UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
GRADO EN INFORMÁTICA

INTERFAZ DE CONSULTAS MAPREDUCE PARA HADOOP

Alumno: Jorge Rivert González
Tutor: Harith Aljumaily
Directora: María Dolores Cuadra Fernández

24 de Febrero del 2015

Resumen

El presente trabajo fin de grado se desarrolla dentro del contexto de Big Data. Este concepto puede definirse como el avance de la tecnología para dar un nuevo enfoque de entendimiento y de la toma de decisiones presentes hoy en día debido al gran crecimiento de datos en las organizaciones. Debido a esto, se entiende que manejar esta gran cantidad de datos sería muy costoso en términos de tiempo y dinero, sin saber si la información recopilada tiene un fin concreto en un primer momento.

La finalidad de este proyecto es realizar un análisis y desarrollo de una aplicación que sea capaz de facilitar el manejo de ficheros tanto en local como en el sistema distribuido de Hadoop, permitiendo así la visualización del contenido y proporcionar una interfaz para realizar consultas MapReduce. La aplicación deberá ser capaz de reconocer los ficheros de texto de manera que puedan ser estructurados en tablas no relacionales, exactamente como distribuye el sistema Hadoop.

El motivo de estudio y desarrollo de este trabajo viene definido por el manejo de grandes volúmenes de datos que pueden almacenarse en un sistema distribuido en el cual no existe ningún tipo de estructura que lo defina. Por ello es necesario una serie de herramientas que permitan visualizar el contenido de los ficheros, transferencia entre local y Hadoop al igual que pueda servir de ayuda para realizar las operaciones más eficientes dentro de los entornos Big Data, como es el MapReduce. De esta manera, la nueva distribución de contenido permitirá analizar de manera más cómoda una información en concreto

Índice de contenidos

Índice de Ilustraciones

Índice de tablas

Glosario de términos

Catálogo de términos específicos del contexto del trabajo.

Hadoop:	Infraestructura digital de desarrollo en Java en código abierto que permite desarrollar tareas muy intensivas de computación masiva, que se dividen en piezas más pequeñas y posteriormente se distribuyen en un conjunto de máquinas.
HDFS:	*Hadoop Distributed File System* es el sistema de archivos distribuido para Hadoop, siendo éste escalable horizontalmente y portátil. Almacena ficheros de gran tamaño a través de muchas máquinas consiguiendo fiabilidad gracias al replicado de datos a través de múltiples hosts
NTFS:	*New Technology File System* es un sistema de archivos que se inició con Windows NT. Su mejora consiste en el tamaño del cluster (a partir de 512 bytes) de forma independiente al tamaño que pueda tener la partición. Se utiliza en estaciones de trabajo y servidores.
NOSQL:	Denominado a una clase de sistema de gestión de bases de datos cuya principal diferencia con respecto a las bases de datos relacionales es que no utiliza el lenguaje SQL, por lo que no es necesario estructuras fijas como las tablas
NameNode:	Pieza central de un sistema de archivos HDFS. Mantiene el árbol de directorios de todos los ficheros del sistema, realizando un seguimiento donde el cluster mantiene los datos del archivo. Es necesario comunicarse con el mismo para interactuar con un fichero (lectura, copia, mover, eliminar…)
DataNode:	Se conecta al NameNode para recibir sus servicios y así disponer de la ubicación de los datos solicitados. Un sistema de archivos funcional tiene más de un DataNode con datos replicados a través de los mismos.
Hadoop Common:	*Es un paquete considerado como la base o núcleo del marco Hadoop. Proporciona servicios* esenciales y procesos básicos como la abstracción del sistema operativo subyacente y un sistema de archivos. Incluye los archivos necesarios para iniciar Hadoop.
HBase:	*Sistema de gestión de base de datos orientada a columnas y que se ejecuta en la parte superior de HDFS. Adecuado para un conjunto disperso de datos y*

	con gran uso.
Java Swing:	*Biblioteca gráfica para Java donde se pueden desarrollar interfaces gráficas con independencia de la plataforma. Utiliza una programación por hilos lo cual otorga extensibilidad al ser una arquitectura particionada. Además es muy personalizable* al representar estilos de apariencia "look and feel".
BI:	*Business Intelligence* o Inteligencia de Negocio es el conjunto de estrategias y aspectos relevantes que están enfocadas a la administración y conocimiento del medio, gracias al análisis de datos existentes en la empresa.
Job Traker:	Es un servicio dentro de Hadoop que realiza MapReduce a los nodos específicos del clúster. Es el encargado de enviar el trabajo a los nodos.
Task Tracker:	Es un nodo del clúster que acepta tareas, mapea, reduce y realiza todo tipo de operaciones aleatorias desde el Job Traker. Cada Task Tracker dispone del número de ranuras que indican las tareas que pueden aceptar.
Datasets:	O colección de datos que normalmente se corresponde con una sola tabla de base de datos o una estadística única de matriz de datos donde cada columna de la tabla representa una variable en particular y cada fila corresponde a un determinado miembro del conjunto de datos.
BigTable:	Sistema de gestión de base de datos creado por *Google* siendo éste distribuido y con alta eficiencia. Almacena la información en tablas multidimensionales y cuyas celdas disponen de versiones temporales de sus valores, por lo que se puede hacer un seguimiento histórico de los mismos.
Data warehouses:	Es una colección de datos orientada a un determinado ámbito y cuyas características son que es integrado, no volátil y variable en el tiempo, ayudando así a la toma de decisiones en la entidad en la que se utiliza.

1 Introducción

Este apartado sirve de guía para presentar el trabajo realizado. Con ello se pretende introducir el proyecto y sus objetivos principales, al igual que los aspectos que han motivado al alumno para su realización final.

A continuación se exponen los propósitos del documento para más adelante detallar el alcance del mismo indicando todos los capítulos de los que consta, incluyendo una breve descripción de cada uno de ellos.

1.1 Definición del problema

La idea principal de este proyecto surge a raíz del problema para visualizar ficheros de texto de una forma totalmente automática en un entorno Big Data y posteriormente generar resultados en forma de listados de tipo clave-valor con el paradigma MapReduce. Con ello se pretende diseñar un entorno visual que permita conocer en detalle la información contenida en un fichero de texto y que debe cumplir una serie de patrones que se comentarán más adelante. Así, el usuario final podrá obtener la información de cualquier fichero de texto que se encuentre en su disco duro y a su vez analizar ese contenido en forma de listados para conocer el alcance de los beneficios que pueda otorgar toda esa información analizada.

Para entender el contexto de trabajo es necesario exponer brevemente la tecnología Big Data, apoyándonos en la ilustración inferior. Esta tecnología está orientada al tratamiento y análisis de gran cantidad de datos, que resultaría imposible poder tratarlos con herramientas de base de datos actuales. Estos datos pueden recopilarse de redes sociales, sensores, imágenes, videos, móviles, GPS, bancos, organizaciones… y son capaces de generar casi 2,5 quintillones de bytes en un solo día [1]. Por ello, es importante tener la capacidad de tratar esta cantidad de datos de manera rápida, si lo que se busca es información casi instantánea.

La ilustración que viene a continuación representa el flujo de información desde que se genera en distintos sistemas hasta que se trata para su análisis en un Data Warehouse.

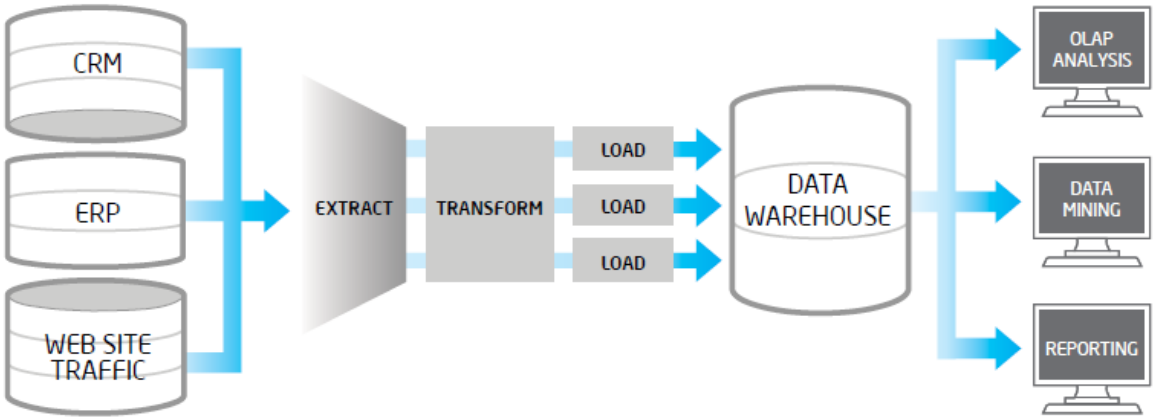


Ilustración 1: Proceso de Big Data [2]

La tecnología Big Data está a la orden del día, donde incluso se han desarrollado máquinas capaces de procesar una cantidad casi descomunal de datos, con procesamiento multiparalelo y tratamiento de datos heterogéneos. Empresas como *Google*, *Yahoo!* o *Facebook* disponen de un almacenamiento de datos basados en esta tecnología que les permite aprovechar todo el potencial de los datos almacenados. Por ejemplo, *Google Analytics* permite conocer estadísticas casi a tiempo real de los usuarios que han visitado nuestra web y a la vez es capaz de mostrar información relevante del sujeto que permita mejorar la calidad de la web para futuros visitantes.

Lo que hoy en día conocemos como Web 2.0 [3] está asociado a todas aquellas aplicaciones web que permiten al usuario crear su propio contenido a la vez que se facilita la compartición del mismo. Es decir, todas aquellas aplicaciones que están enfocadas al usuario final tales como Blogs, Wikis, Redes sociales,…

Esta últimas son el gran medio de comunicación social de nuestra época, centradas básicamente en relaciones on-line, como amistades, comunicación social o compartición de sensaciones, sentimientos e incluso ficheros entre usuarios. Algunos ejemplos de este tipo de redes son *Google+*, *Whatsapp*, *Twitter* o *Facebook*.

En la actualidad existen pocas herramientas y conocimientos aplicados a la tecnología Big Data. Es necesario dar un empuje en forma de beneficios que puede otorgar esta tecnología ya que las organizaciones no creen necesario su implantación de momento.

Pero realmente se están perdiendo grandes volúmenes de datos que pueden beneficiar a la empresa, como mejoras en sus productos, atención personal al cliente o cualquier tipo de problema que pueda surgir es posible tratar con Big Data y obtener información que de otra forma no se hubiera podido.

El problema reside en el sistema de ficheros distribuido propio de Hadoop a la hora de decantarse por esta tecnología para una empresa, debido principalmente a su ignorancia o falta de información al respecto. Este punto es uno de los grandes problemas de la tecnología Big Data. Recordemos que Hadoop es una distribución Big Data para trabajar con grandes volúmenes de

datos, por lo que hoy entendemos como manejo de datos en un sistema de ficheros NTFS [5] resulta inservible para tal cantidad de datos.

Surgió la idea de particionar los ficheros en bloques para poder trabajar en paralelo sobre un mismo fichero y si a ello le sumamos un paralelismo en cuanto a nodos se refiere obtenemos una velocidad de lectura realmente alta y que es justo lo que se necesita para manejar un fichero de enormes proporciones. Todo este sistema de ficheros y su funcionamiento se denomina HDFS [4] y es una tecnología que hoy en día está patentada por Apache Hadoop, por lo que su manejo queda anclado a esta tecnología. El problema es que es necesario tener conocimientos de la materia para entender el funcionamiento de este sistema de ficheros y será uno de los puntos clava cuando comience el desarrollo del proyecto.

Pocas herramientas en el mercado pueden proporcionar consultas MapReduce con tecnología Hadoop, por lo que se trata de dar al usuario final una herramienta que sea capaz de trabajar con ficheros de alto contenido y tamaño, manejar esos ficheros con el sistema propio de ficheros de Hadoop, ser capaz de analizar su contenido y/o estructura y también realizar consultas MapReduce de la manera que ellos deseen.

Se cree que las aportaciones de esta aplicación con respecto a otras del mercado son bastante diferentes como para decantarse por un software u otro, siempre teniendo en cuanta la finalidad del proyecto y que el usuario esté buscando exactamente eso.

1.2 Objetivos

A continuación se exponen los objetivos del presente trabajo fin de grado a modo explicativo. Se intenta dar un punto de vista antes de comenzar con el tratamiento de toda la información recopilada en los siguientes capítulos, por lo que se dice que es un avance de los mismos.

- Estudiar el ámbito de las tecnologías en las que se desarrolla el TFG, como son Big Data y Hadoop.

- Estudiar y analizar las herramientas actuales en el mercado de la comunicación con HDFS-Hadoop para así conocer los nuevos aportes que puede otorgar la nueva aplicación a desarrollar con respecto a las otras existentes en el mercado

- Diseñar e implementar una herramienta que tiene como tareas:

    - Facilitar el intercambio de ficheros entre local y HDFS para proporcionar una aplicación final que permita tener el control total de ficheros HDFS de la misma forma que se manejan ficheros en la plataforma Windows.
    - Lectura de ficheros de texto y visualización del contenido a modo de esquema tanto parcial como total.

  - Proporcionar un interfaz de consultas MapReduce para ayudar a los usuarios “con poca experiencia en la implementación” a someter distintas consultas multiclave/multivalor en Hadoop [8].

- Realizar pruebas de evaluación cuya finalidad es demostrar que los puntos a desarrollar se han cumplido y que la aplicación realiza todas las funcionalidades que se han explicado en los anteriores puntos.

1.3 Entorno socio-económico y marco regulador

La tecnología Big Data permite el procesamiento de grandes volúmenes de datos, con el propósito de obtener información y generar conocimiento a partid de ellos.

El volumen de información disponible, la variedad que pueda existir de datos y la velocidad con la cual se procesan, aportan una gran cantidad de posibilidades sin precedentes. Pero todavía existen muchas dudas en las empresas que todavía no son conscientes de los desafíos normativos que se pueden generar con el tratamiento de datos de los propios usuarios.

Estos desafíos han sido analizados desde un punto de vista legal para poder fijar una estrategia de cumplimiento. Aspectos como la seguridad, privacidad y conservación de datos han de tenerse en cuenta cuando hablamos del marco regulador en Big Data, al igual que los derechos de los propios consumidores. Además no solo debe asegurarse la privacidad de los datos, sino que también se debe asegurar la transparencia en la utilización de los mismos.

Actualmente el procesamiento de esta información en forma de datos que requiere Big Data se enmarca dentro de la Ley Orgánica 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal, lo cual implica un anonimato total de aquellas personas que pertenecen a los datos citados.

Estos datos deben ser usados dentro de una forma legal por parte de terceras personas autorizadas para cumplir la legislación legal. Por lo tanto, el procesamiento de estos datos debe ser anónimo y se debe cumplir la privacidad de los usuarios.

1.4 Propósito del documento

Este documento se realiza con el fin de exponer el trabajo realizado durante todo el desarrollo del trabajo fin de grado, pasando por todas las fases del mismo como recopilación de diferentes tecnologías, diseño, implementación o pruebas realizadas.

Por así decirlo, la finalidad del mismo es demostrar con hechos los resultados del proyecto para que el lector pueda entenderlos e interpretar el trabajo realizado.

Dado que es necesario investigación previa de la tecnología a utilizar, se puede decir que el trabajo conlleva una gran carga en cuanto a tratamiento de la información se refiere ya que será necesario conocer diferentes herramientas que utilizan la misma tecnología y como se puede despuntar en ese aspecto.

1.5 Fases del desarrollo

- Estado del arte
  En esta fase se estudia el contexto en el que se enmarca el problema que se debe resolver. En ella se examinarán las principales herramientas disponibles en el mercado sobre Big Data y las diferentes tecnologías a tratar para así poder seleccionar lo más adecuado para desarrollar el proyecto.

- Definición del problema
  Esta fase debe examinar cómo trabaja Hadoop en entornos Big Data y cómo se podría tratar la información de un documento de texto para que sea útil, estableciendo así el problema a tratar.

- Solución al problema
  La solución se desarrolla a lo largo de esta etapa, donde se debe identificar las características esenciales y las detalladas del mismo.

- Evaluación
  Una de las etapas más importantes de cara a la calidad del producto final es la etapa de evaluación. Se debe llevar a cabo una evaluación para poder así demostrar la validez del software implementado y que cumpla con la solución al problema inicial. Además, se debe demostrar las características del anterior apartado.

1.6 Medios empleados

Los principales recursos utilizados para desarrollar e implementar la solución al problema se pueden dividir principalmente en dos categorías: Fuentes bibliográficas y herramientas necesarias para implementación de la aplicación.

La primera proporciona todos aquellos conocimientos necesarios para elaborar una solución óptima al problema, como por ejemplo las tecnologías que actualmente se implementan para casos similares. A continuación se exponen las fuentes bibliográficas que más se han utilizado durante esta fase:

- Documentos PDF
- Biblioteca Carlos III
- Revistas tecnológicas
- Herramientas actuales con funcionalidades similares a la que se quiere desarrollar
- Internet

El segundo recurso utilizado se aplica para el desarrollo de la solución, herramientas necesarias para su implementación y el entorno de trabajo utilizado. Este punto es importante para el desarrollo de la implementación del código y son necesarias herramientas específicas para ello:

- **Windows Server 2008 R2:** Como sistema operativo en el cual se instalan las herramientas necesarias para el desarrollo de la aplicación
- **Eclipse Kepler:** Es una herramienta de código abierto multiplataforma que permite el desarrollo en Java de aplicaciones en entornos de desarrollo integrado.
- **Apache Hadoop:** Framework que soporta aplicaciones distribuidas. Permite trabajar con grandes volúmenes de datos y miles de nodos. Está inspirado en *Google* y su MapReduce
- **Java Swing:** Es una biblioteca gráfica para Java que permite desarrollar interfaces gráficas para el usuario

1.7 Estructura de la memoria

Esta memoria consta de ocho capítulos:

- **Introducción al trabajo realizado y contenido de la memoria**

- **Estado de la cuestión:** Se establece el contexto del problema a tratar, se revisan las principales tecnologías para desarrollar un entorno Big Data que existan en la actualidad y las distintas formas de manejar su contenido de tal manera que resuelva el problema a tratar.

- **Gestión del proyecto software:** Se refleja el alcance del proyecto, el plan de trabajo adoptado, la gestión de los recursos necesarios para el desarrollo del mismo, los posibles riesgos y el plan de pruebas.

- **Solución:** Se debe detallar como se ha llevado a cabo la solución, donde se describe todo el proceso del desarrollo para llegar hasta este punto.

- **Evaluación:** Cuando se dispone de la solución será necesario evaluarla, demostrando así su validez ante el problema. Será necesario llevar un control sobre las pruebas realizadas y si cumple con los objetivos expuestos desde un principio.

- **Conclusión:** Se proporcionará un resumen del trabajo realizado y una visión personal de la misma.

- **Bibliografía:** Se exponen todas las referencias utilizadas durante el desarrollo del proyecto.

- **Anexos:** Posibles anexos para no interferir en la lectura esencial del documento y que sirven como apoyo a la lectura del mismo

2 El estado de la cuestión

En este apartado se estudia el contexto de trabajo que comprende a este proyecto. Por lo tanto, en este capítulo se revisarán las diferentes técnicas que dan soporte a un entorno HDFS de datos de Hadoop.

El primer apartado valora las diferentes tecnologías que existen en la actualidad para desarrollar un entorno Big Data, y que en este caso se centrará mayoritariamente en Hadoop.

2.1 Tecnologías Big Data

El Big Data o grandes volúmenes de información [9], son conjuntos de datos que crecen con el tiempo de una manera casi exponencial. Debido a ello, resulta bastante incómodo poder trabajar con las herramientas de gestión de bases de datos tradicionales. Estas dificultades se deben esencialmente a la captura, el almacenamiento, la búsqueda de información, el intercambio de datos, posterior análisis y la visualización de los datos.

Big Data utiliza un tipo de tecnología heterogénea, que es aquella donde sitios diferentes utilizan diferentes sistemas de gestores de datos (*DBMS*), y donde cada uno es autónomo. Es decir, debe haber una sincronización entre gestores diferentes de manera que puedan cooperar en el procesamiento de transacciones, ya que cada base de datos dispone de tipos y formatos totalmente diferentes entre sí.

Ilustración 2: Transferencia de datos de las empresas más importantes en la actualidad [11]

Entre las ventajas que podemos encontrar en un sistema heterogéneo está la potencia que ofrece diferentes computadoras conectadas en red, la perfecta integración entre sistemas y dispositivos o que las conexiones entre computadoras no generan ningún problema.

Esta solución puede decirse que es altamente escalable y con una gran potencia acorde a la flexibilidad (que evitará problemas de falta de recursos). Además, se puede decir que nunca quedará obsoleta con el tiempo ya que es posible modificar las características de cada componente.

2.1…1 Requisitos Big Data y Tipos de datos a utilizar

Cuando se debe implementar una tecnología Big Data para el problema a resolver, debe quedar claro si la solución propuesta con las herramientas es lo suficientemente óptima como para costear todo lo que necesita esta infraestructura.

Para ello se elaboran una serie de preguntas que la empresa debe tener en cuenta cuando se procede a implementar una solución al problema mediante técnicas de Big Data [10]:

- ¿Es la solución escalable a grandes volúmenes de datos?
- ¿Permite la solución el procesamiento de flujos de datos?
- ¿Se aprovecha las ventajas del cómputo distribuido?
- ¿Se permite realizar un análisis rápido de los datos?
- ¿La solución permite visualizar los datos y está preparada para integrarse con otros proveedores de tecnología?

Si se obtienen las respuestas a todas estas preguntas, será un éxito la iniciativa Big Data.

En cuanto a los tipos de datos, se disponen de los siguientes:

- **Estructurados:** Este tipo de datos son los conocidos para almacenar y organizar datos en un ordenador de manera eficiente. Disponemos de los tipos *int, boolean, chart,…*
- **No estructurados:** Aquella información que no tiene un modelo de datos definido y no es organizado de ninguna forma predefinida.
- **Semiestructurados:** Son aquellos que no residen en bases de datos relacionales, pero su organización interna permite un fácil tratamiento de los datos.

Cuando una empresa está inversa en la tecnología Big Data empiezan a surgir las dudas de cómo tratar la información y los retos que existen en la actualidad para obtener una solución a corto plazo. La ilustración3 muestra los retos actuales.

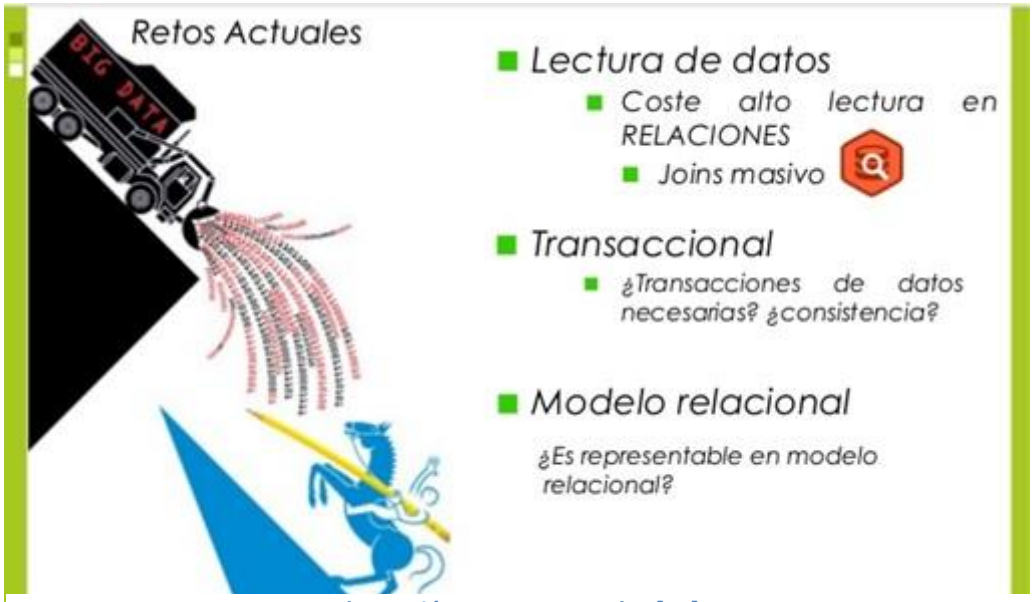


Ilustración 3: Retos actuales [11]

2.1…2 Schema on read vs schema on write

Para conocer la arquitectura de un entorno Big Data se debe comparar con la tecnología actual de bases de datos. Estos dos tipos de arquitecturas representan el presente y el futuro, el cual está más ligado a Big Data [7]

- **Schema on write:** Se define como la arquitectura base para las bases de datos relacionales que conocemos como SQL. Necesita que la estructura del objeto esté previamente definida antes de cargalo, mientras que los datos deben ser validados contra esa misma estructura.
- **Schema on read:** Este tipo de arquitectura no define los objetos previamente a la carga de los mismos, por lo que no existe validación de los datos contra la estructura. Esto es posible gracias a que la estructura de la base de datos se define en la lectura, por lo que es posible cambiar la estructura base de los datos al obtener cada tipo de lectura.

A continuación se exponen dos ejemplos entre las distintas arquitecturas:

- **Schema on write:**

1. Creamos el schema:

   create table clientes (cod_cliente int, des_cliente varchar(50), ...)

2. Volcamos los datos:

   bulk insert cliente from 'C:\temp\clientes.txt' with delimitier = ';'

3. Leemos los datos:

   select cod_cliente, descliente from clientes

- **Schema on read:**

1. Volcado de datos:

   hdfs dfs - copyFromLocal /temp/clientes.txt

   /user/hadoop/clientes

2. Lectura de datos:

   hadoop jar Hadoop-streaming.jar

   -mapper clientes-mapper.py

   -reducer clientes-reducer.py

   -input /user/hadoop/clientes/clientes.txt

   -output /user/hadoop/output/query1

Si se observa el esquema *on read* primeramente se realiza un volcado de los datos al sistema de ficheros HDFS de clientes.txt para luego realizar una lectura de fichero mediante MapReduce. Es necesario disponer del codigo mapper y reducer del paradigma en diferentes ficheros, indicando además el fichero de entrada y la ubicación del fichero de salida con los resultados obtenidos una vez lanzado MapReduce.

2.1…2.1 Ventajas e inconvenientes de cada tipo de arquitectura

Debemos conocer el alcance de cada arquitectura, cual es nuestro problema a tratar y considerar que estructura es la que se debe adoptar para alcanzar la meta. El uso de uno u otro esquema viene determinado por los tipos de datos que se deben almacenar o consultar, pero sobre todo de la capacidad de proceso con la que contamos.

Dado que Big Data necesita de una capacidad de procesamiento de datos muy alta, no tendría sentido utilizar este tipo de tecnología, siendo mas recomendable una base de datos tradicional.

Aun así, es recomendable conocer las posibles ventajas e inconvenientes que reflejan estas dos arquitecturas.

- **Schema on write:** La base de datos relacional tiene como principal ventaja la relacion entre entidades y además es posible mantener los datos documentados en la propia estructura. Ésto es posible gracia a que las estructuras estan predefinidas en un primer momento.
  La gran desventaja de esta arquitectura son los tiempos en las validaciones de carga de los datos, ya que son penalizados pero tiene como consecuencia una lectura mucho mas rápida. Es decir, siempre sufrirá los altos tiempos de volcado de datos a la estructura.

- **Schema on read:** Este tipo de arquitectura no realiza validaciones en la carga de datos por lo que su volcado es mucho más rapido. En este caso las penalizaciones se las llevará las lecturas ya que es ahí donde se realiza la estructura.
  La mejora que otorga este esquema es la alta flexibilidad en la lectura de los datos según la vista que más nos interese en cada tipo de consulta a realizar, pero por el contra no existe una documentacion de los datos como tal.

  Podemos indicar que este tipo de arquitectura está diseñada para aquellos datos no estructurados, donde se puede guardar además el nivel de detalle más atómico, pero necesita una capacidad de procesamiento muy alta para rendir.

1.1...3 Tipos de bases de datos Big Data

El coste tan elevado de lectura de datos con bases de datos relacionales provocó como respuesta el paradigma *NoSQL*. Es un sustituto de las bases de datos relacionales que busca en escenarios específicos otras opciones, otorgando así un abanico de opciones al desarrollador. Se disponen de los siguientes tipos en el mercado:

- *Almacenes Key-Value*

Este tipo de base de datos es mucho más simple en cuanto a su uso, donde el valor guardado se almacena como un array de bytes (*BLOB*). La gran diferencia es que la base de datos no tiene constancia de la importancia del contenido, sino que la clave y el valor asociado a esta son lo único que tiene como relevante.
Además, no será necesario definir un esquema previo para almacenar la información. Es decir, el número de columnas y el tipo de dato asociado no estarán definidos.

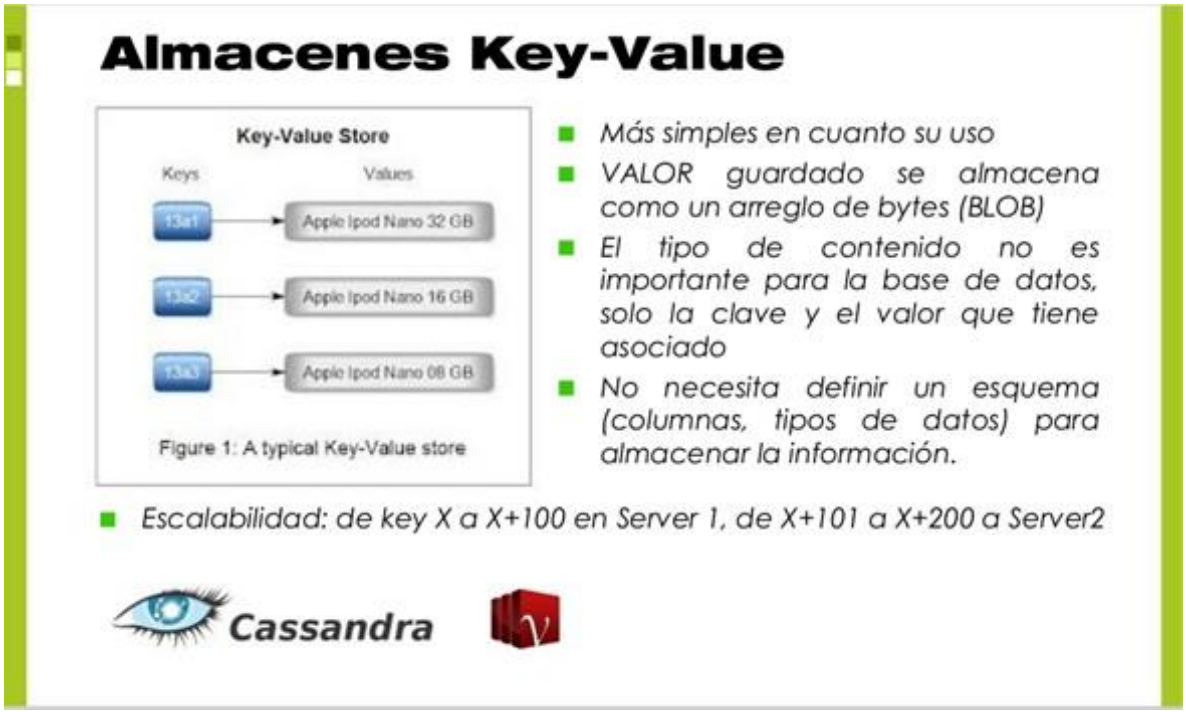


Ilustración 4: Almacenes Key-Value [11]

- *Bases de datos orientadas a documentos*

Se puede decir que son prácticamente idénticas a los almacenes *key-value [12]*, con la única excepción de que el valor no se guarda como campo binario, sino con un formato que el propio servidor pueda entender. En este caso, la base de datos no puede entender el formato. Entre las más importantes se puede encontrar:

- *CouchDB*
- *MongoDB*
- *RavenDB*
- *BaseX*
- *eXist*
- *IBM Lotus Dominio*

Ilustración 5: Bases de datos orientadas a documentos [13]

- *Bases de datos orientadas a grafos*

Los datos se almacenan en forma de grafos, dando importancia a la relación existente entre los datos. Así es más eficiente navegar entre las relaciones. Si la información disponible para almacenar se puede representar fácilmente como una red, este tipo de base de datos es la recomendada.

- *Bases de datos columnares*

De entre todas las bases de datos mencionadas, *HBase* utiliza ésta, que a su vez es recomendada como la mejor para tratar *data warehouses* y sistemas de *Business Intelligence*. En este caso se almacena la información en columnas en vez de filas como se realiza en una base de datos relacional.

Este pequeño detalle otorga una mayor rapidez en las lecturas, pero no es eficiente en las escrituras. Son ideales para calcular datos agregados

Ilustración 6: Bases de datos columnares

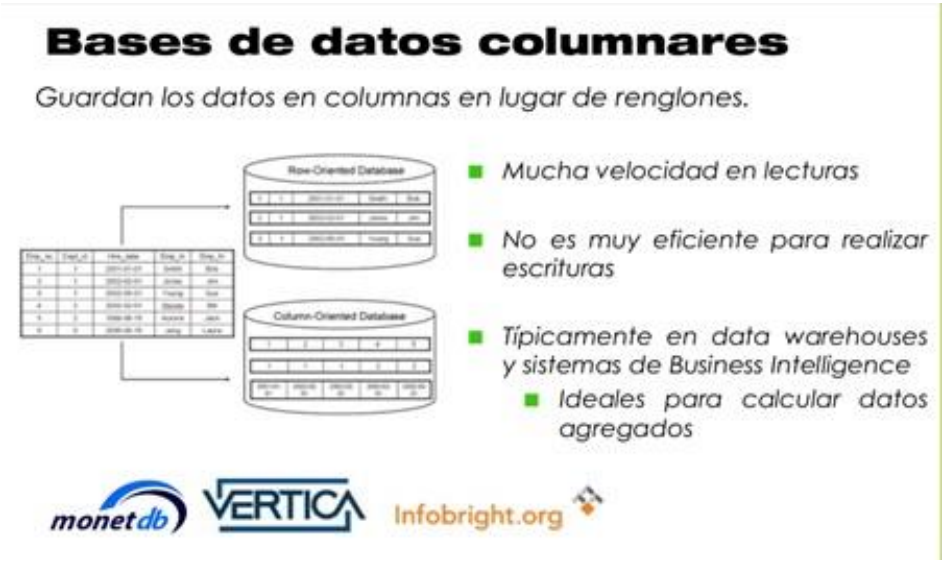


Ilustración 6: Bases de datos columnares [14]

1.1…1 Apache Hadoop

Es un proyecto de código libre desarrollado por la empresa *Apache* [6] junto con la colaboración de importantes empresas como *Microsoft, Yahoo!* o *Facebook*. Su principal finalidad es consolidar un conjunto de herramientas java el desarrollo de sistemas escalables y distribuidas. Además posee *frameworks* propios otorgando una estructura única para Big Data.

Los cimientos de Hadoop se basan en el MapReduce de *Google* (que más tarde se explicará) al igual que el *Google File System* para su sistema de ficheros distribuido. Para tener una idea clara de la gran cantidad de empresas que ya están utilizando esta tecnología se exponen las más importantes*: eBay, Google, Spotify, Microsoft, Facebook, Yahoo!, Indra* o *Twitter*.

Como se ha comentado anteriormente, Hadoop es una herramienta para procesar gran cantidad de información distribuida a través de un modelo de programación que resulte sencillo para el usuario medio. Está diseñado para trabajar de forma escalable para así poder funcionar con clústeres de un solo nodo al igual que con cientos de ellos. La gran ventaja que otorga Hadoop es que es posible distinguir los nodos que dan error para ofrecer una gran tolerancia a errores.

Su éxito radica en su sencillez y funcionalidad, por ello se está hablando de la capa inferior más utilizada para MapReduce

Hadoop se basa en dos módulos claramente diferenciados:

- **Hadoop Distributed File System o HDFS:** Es el sistema de ficheros que utiliza Hadoop para ejecutar la gran mayoría de herramientas disponibles en su ecosistema.
- **Hadoop MapReduce:** Es la base de Hadoop, el framework de programación para el desarrollo de aplicaciones.

Existen diferentes versiones de Hadoop para distintas distribuciones, y concretamente se está utilizando la 2.0 en la actualidad, por lo que se expone esta versión que es la que se ha estado implementando en el TFG.

- *Versión 2.0 Hadoop* [15]

Esta versión sigue coexistiendo con la hermana inferior y es utilizada por muchos desarrolladores al ser bastante más estable que la predecesora. Sus pilares fundamentales se basan en tres bloques claramente diferenciados:

- **HDFS:**

Como se ha comentado anteriormente es el sistema de ficheros distribuido sobre el cual se ejecutan todas las herramientas Hadoop, al igual que proporciona un abanico de características al propio sistema. Su principal aliciente es que es posible utilizando en la mayoría de los sobremesas de coste asequible al igual que posee una tolerancia a fallos.

Entre sus características más reseñables podemos indicar que está diseñado para parecerse a un **sistema de ficheros convencional**, para así familiarizarse rápidamente con el mismo. Para los usuarios *Unix*, podemos indicar que es prácticamente una copia, desde los permisos de ficheros a los de seguridad.

Otra de las características más importantes es la **tolerancia a fallos**. Normalmente estamos hablando de una gran cantidad de servidores trabajando al mismo tiempo, por lo que los fallos deben quedar almacenados en cada una de las partes del sistema de ficheros, lo que implica que un fallo de hardware no es la excepción.

A continuación el resto de características [16]:

Streaming: Se proporciona al usuario un acceso a los datos con un rendimiento elevado.

Portabilidad: Es necesario que el sistema sea portable para gran cantidad de plataformas (hardware y software)

Escalabilidad: Aunque no llegue a ser una escalabilidad avanzada, HDFS permite la expansión de forma sencilla en caliente, pudiendo añadir nuevos nodos si la necesidad de parar los procesos en ejecución en el propio clúster. Es el propio sistema el que se encarga de determinar los bloques de fichero que serán almacenados y que trabajos tendrá que realizar.

Gran cantidad de información: Los datos almacenados deberán disponer de una capacidad total del sistema acorde a ellos. Un tamaño medio de un fichero HDFS está pensado en gigabytes hasta los terabytes.

Todo esto es muy útil, ¿pero cómo actúa por dentro HDFS?

Para aclarar este tema, tenemos que centrarnos en la arquitectura utilizada por HDFS. En este caso estamos hablando de arquitectura maestro-esclavo. Se compone de un *NameNode* como

función maestro, permitiendo coherencia entre el sistema de ficheros. Además de ser un servicio único permite también el acceso a diferentes clientes sobre un mismo fichero.

El servicio que actúa como esclavo es el llamado *DataNode*, el cual siempre está operativo en todos los nodos de un clúster. Su finalidad es administrar en todo momento el almacenamiento de los datos en el nodo donde se esté ejecutando.

Este sistema de ficheros tiene la particularidad de almacenar cada fichero en diferentes bloques y a su vez, cada bloque se divide en un nodo distinto. Así, es más cómodo trabajar con el algoritmo MapReduce, pudiendo acceder a diferentes partes de un mismo fichero de forma totalmente paralela, lo cual otorga una velocidad mayor en la lectura de ficheros.

Dado que es posible que surja cualquier error en algún nodo, HDFS está preparado para asegurar la disponibilidad y evitar la pérdida de ficheros mediante la replicación de nodos.

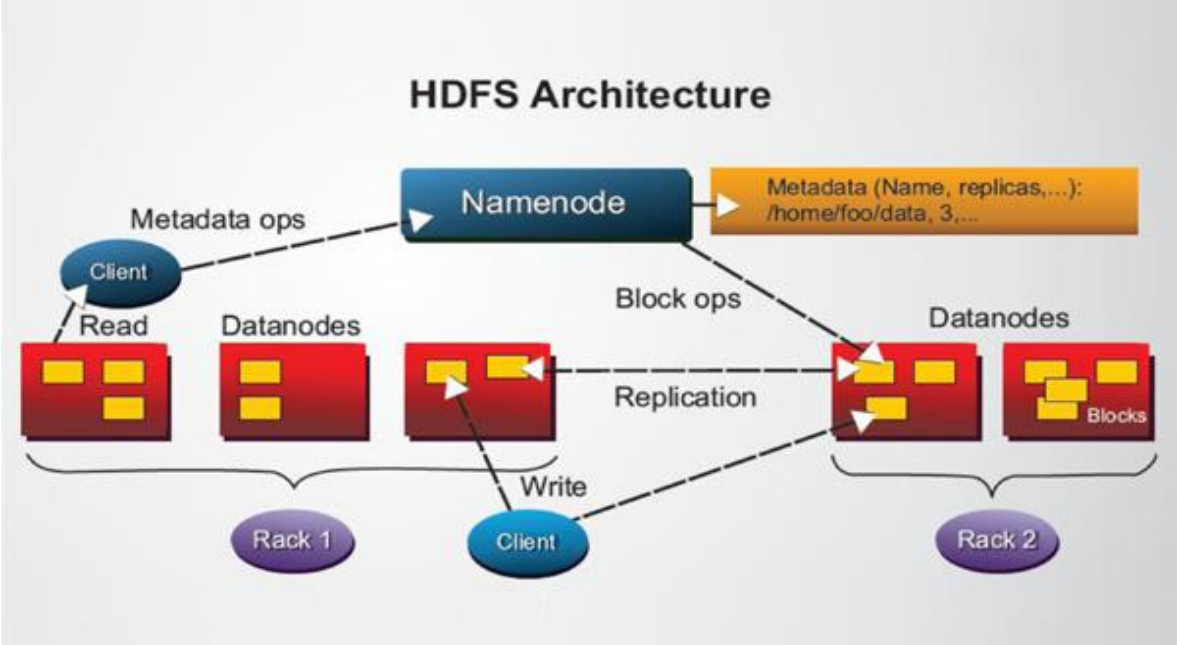


Ilustración 7: Arquitectura HDFS [11]

HDFS también nos brinda la oportunidad de configurar el tamaño de bloque, siendo 128MB lo más aconsejable. Este tamaño parece excesivo si tenemos en cuenta que es solo una parte del fichero, pero recordemos que los ficheros manejados por Hadoop tienden a ser de tamaños bastante grandes.

Otro de los apartados de configuración en el sistema de ficheros es el factor de replicación. Este número indica las veces que debe replicarse un mismo fichero, siendo recomendable un mínimo de 3 pero no se especifica el máximo. Hay que tener en cuenta que un factor de replicación alto repercute seriamente al rendimiento del equipo y por consiguiente de la consulta a realizar sobre el fichero, nunca olvidando que el espacio en el disco sería mucho mayor.

Hadoop Common es la herramienta encargada de proporcionar el acceso a los sistemas de archivos que soporta *Hadoop*. Para disponer de una programación efectiva de trabajo es necesario que cada sistema de archivos reconozca y proporcione su ubicación. Esta información permite a las aplicaciones Hadoop ejecutar trabajo en el nodo donde están los datos para reducir el tráfico de la red.

Entonces, ¿cuál es la finalidad de Hadoop? Proporcionar una compresión de datos mediante el procesamiento paralelo de muchos nodos. Así, las búsquedas, el procesamiento de logs de información, la analítica en *Facebook* o *Linkedin* y la retención de datos son el uso mayoritario de esta herramienta.

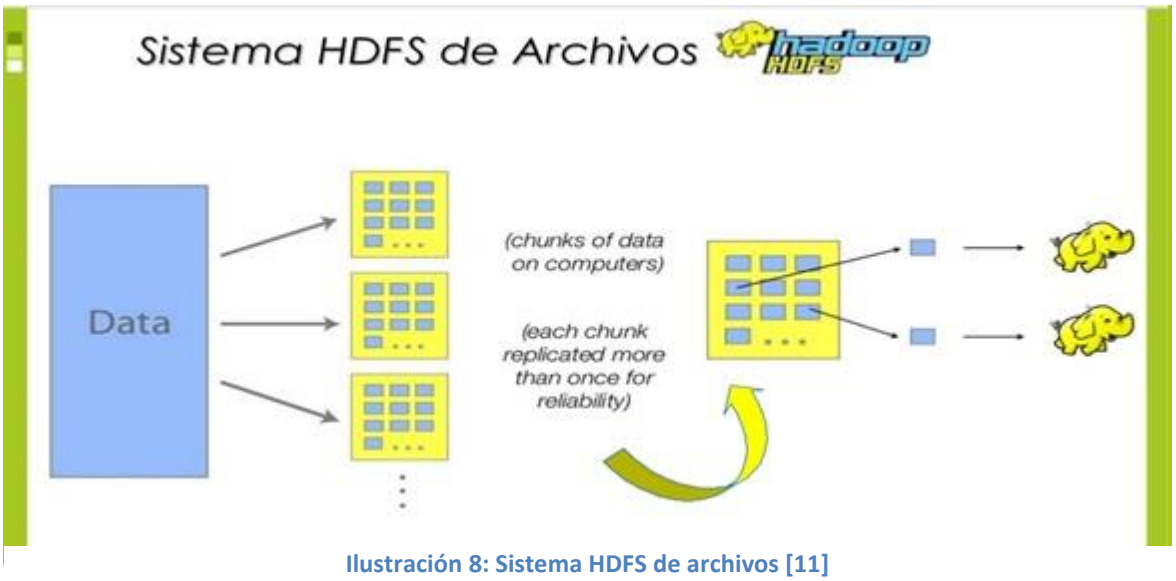


Ilustración 8: Sistema HDFS de archivos [11]

- **MapReduce**: Es un rastreador de trabajos o Job Traker que impulsa el trabajo fuera de los nodos Task Tracker disponibles en un clúster.

Este modelo de programación es el que utiliza actualmente *Google* para la computación paralela sobre grandes colecciones de datos en distintos grupos de máquinas. Se emplea para resolver algunos algoritmos susceptibles de ser paralelizados, pero no es la solución a cualquier problema. En general se utiliza para abordar problemas con *datasets* de gran tamaño.

No todos los procesos pueden ser abordados por MapReduce, únicamente aquellos que se pueden dividir en operaciones de *Map()* y de *Reduce()*. Ambas están definidas con respecto a datos estructurados en tuplas (clave, valor).

Función Map(): Se toma uno de estos pares de datos con un tipo en un dominio de datos y devuelve una lista de pares en un dominio diferente.

$$Map(k_1, v_1) \rightarrow list(k_2, v_2)$$

Ilustración 9: Función Mapper

Esta función es la encargada del mapeo y es aplicada en paralelo para cada ítem en la entrada de datos. A la salida se dispone de una lista de pares (list(k2,v2)) por cada llamada. A continuación, MapReduce() junta todos los pares con la misma clave de todas las listas y los agrupa, creando así diferentes grupos por cada clave generada.

Función Reduce(): Esta función recoge los datos obtenidos de la fase *Map* y se ordenan de tal manera que los pares clave-valor sean contiguos, aplicándose en paralelo por cada grupo que se haya creado anteriormente.

La salida generada se obtiene al aplicar una función a todos los pares recibidos con la misma etiqueta, así, la función sería la siguiente:

Reduce (clave2, lista(valor2)) –> lista(valor2)

Ilustración 10: Función Reduce

La función final del MapReduce() debe obtener unos resultados completamente ordenados por pares clave-valor.

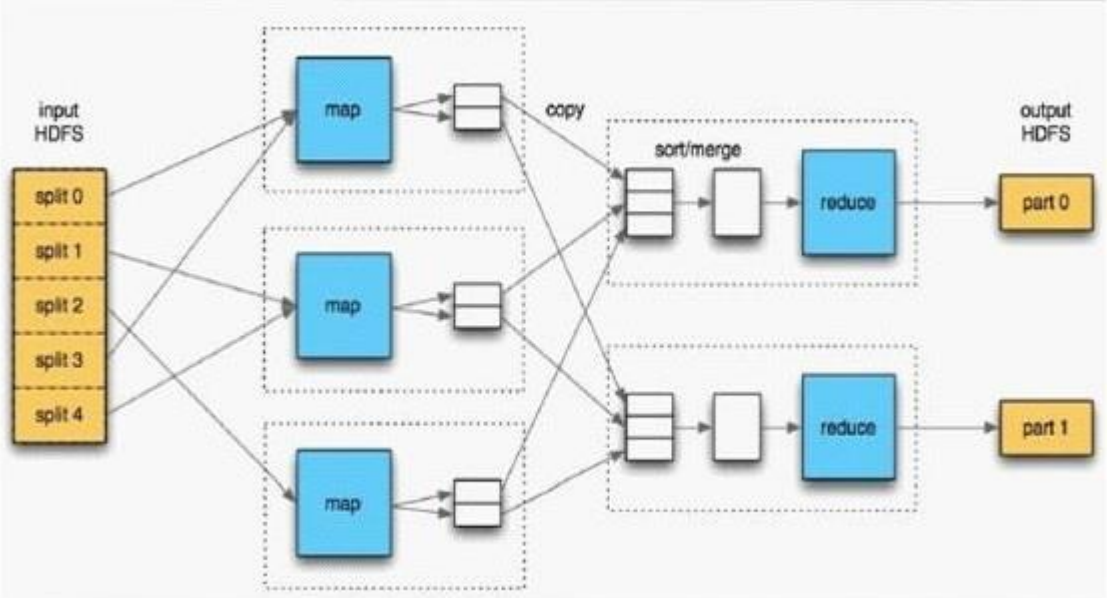


Ilustración 11: Proceso MapReduce

1.1...1 Apache HBase

También desarrollado por *Apache* e implementado en lenguaje Java, *HBase* es una base de datos abierta y distribuida que implementa un concepto importante: BigTable.

Este sistema de gestión de base de datos fue desarrollado por *Google* en el año 2004. Posee la capacidad de almacenar la información en tablas multidimensionales, donde la mayoría de las celdas están vacías. Además estas celdas tienen "memoria" que permite recordar los valores históricos almacenados en ellas.

Cuando se quiere consultar la base de datos, ésta trabaja de tal manera que las tablas se dividen en columnas y son almacenadas con un tamaño de entre 100 y 200 Mb. El sistema de balanceo desarrollado por *Google* permite a la maquina desprenderse de algunas tablas que no se estén utilizando y trasladarlas a otra máquina para acelerar el proceso de búsqueda en una tabla en concreto, pudiendo incluso realizar una rápida recomposición del sistema si una maquina se "cae".

HBase permite escalar casi linealmente simplemente agregando más servidores al sistema, permitiendo además almacenar gran cantidad de tablas de un tamaño enorme (miles de millones de filas y columnas).

Por lo tanto, este tipo de base de datos es la más apropiada en entornos Big Data gracias a sus pequeños tiempos de acceso en lectura y escritura y su capacidad de almacenamiento.

Ilustración 12: Apache HBase

1.1…2 Apache Cassandra

Es otra base de datos de tipo *NoSQL* basada en el modelo de almacenamiento clave-valor.

AL igual que *HBase*, esta herramienta permite almacenar grandes volúmenes de datos en forma distribuida. Entre sus objetivos principales están la completa escalabilidad lineal y la disponibilidad.

Su funcionamiento se basa en una serie de nodos iguales que se comunican entre sí mediante un protocolo P2P, siendo la redundancia máxima en este caso.

Actualmente esta base de datos es utilizada por *Twitter* para almacenar todos los tweets e información relevante que consideren.

2.2 Aplicaciones actuales con tecnología Hadoop

Este apartado habla sobre las tecnologías actuales que utilizan Hadoop para su funcionamiento, principalmente desarrolladas para trabajar más cómodamente con grandes volúmenes de datos. Repasaremos las principales y las que más se puedan ajustar al modelo que se tiene en mente para el desarrollo del TFG.

2.2.1 HDFS Explorer

Su principal cometido consiste en dar soporte al usuario final para manejar grandes volúmenes de datos. Si tenemos en cuenta la gran cantidad de información que se puede recoger de cualquier sistema o herramienta, tenemos el problema del manejo de los mismos dado que se está hablando de millones de registros en un minuto, por ejemplo.

La herramientas actuales, a excepción de HDFS Explorer, no proporcionan un manejo simple de estos datos al usuario final, por lo que se ha desarrollado una herramienta que imita al sistema de ficheros más estandarizado hasta la fecha: *NTFS*, lo que comúnmente se conoce como el sistema de ficheros de Windows.

Manejar los clusters, nodos y direccionamiento de ficheros son tareas muy complicadas si el usuario no está familiarizado con Hadoop, por lo que la idea principal es minimizar el coste de estas tareas para mostrar al usuario un entorno sencillo y perfectamente comprensible, de tal manera que pueda manejar ficheros entre Hadoop y su máquina.

Entre sus principales características encontramos:

- Carga y descarga de ficheros
- Manejo de ficheros: copiar, mover, eliminar y renombrar
- Integración total con Windows
- La manera más cómoda para tratar ficheros HDFS y alojamiento en Windows
- Diseñado para HDFS: Conexión con el sistema de ficheros casi instantánea
- Dirección de múltiples clusters en Hadoop

Características adicionales:

- Integración con Hortonworks Sandbox y Clouder Quickstar
- Soporte de múltiples ventanas

Esta herramienta está desarrollada por *Redgate*

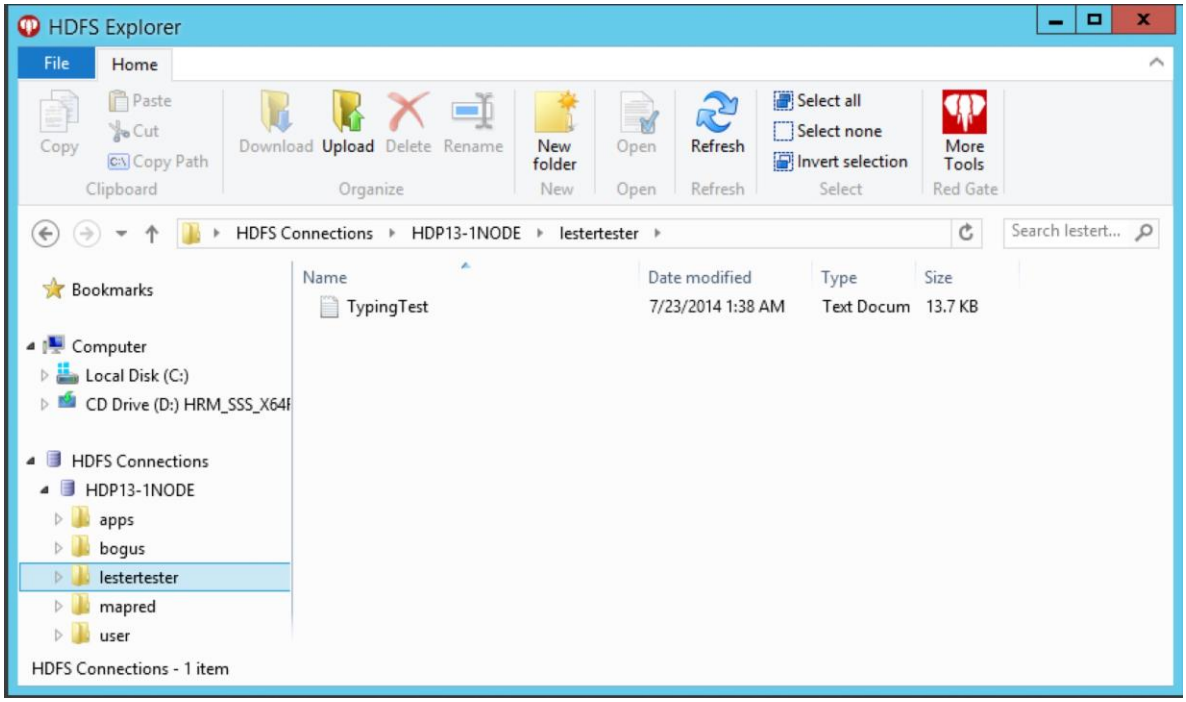


Ilustración 13: HDFS Explorer

2.2.2 Avro

Este sistema de serialización de datos permite transportar información entre herramientas y aplicaciones Hadoop.

Entre sus principales características provee a las aplicaciones que la utilizan de un formato de serialización binario, con un sistema de transporte efectivo y a la vez rápido, escritura de datos de forma persistente con ficheros, implementación del protocolo de maestro-esclavo o RPC (Remote Procedure Call).

Para entender el funcionamiento de Avro se debe tener claro el esquema *JSON [17]*:

```
{"namespace": "ejemplo.avro",
"type": "record",
"name": "Usuario",
"fields": [
        {"name": "nombre", "type": "string"},
    {"name": "numero_favorito", "type": ["int", "null"]},
        {"name": "color_favorito", "type": ["string", "null"]}
 ]
}
```

Ilustración 14: Ejemplo esquema JSON

Cuando los datos sean leídos, la aplicación destino dispondrá de toda esa información necesaria para interpretarlos. La ventaja radica en que siempre se dispone del esquema de datos junto con toda la información serializada cuando la aplicación destino espera recibir otro tipo de datos o esquema distinto al obtenido. Es ahí donde la aplicación puede decidir qué pasos realizar.

2.2.3 Harnessing

Esta herramienta está pensada para la inteligencia de negocio, orientada a los usuarios de tal manera que puedan utilizar grandes volúmenes de datos útiles para su negocio.

Harnessing permite representar de manera inteligente solo los datos útiles o relevantes de manera contextual siempre orientados al problema en cuestión. Por ejemplo, un ejecutivo puede estar interesado en datos relevantes de una línea de productos de su empresa, mientras que un gerente de un producto necesita más datos relevantes en detalle, pero únicamente en esas áreas que realmente la persona supervise.

Entre sus puntos fuertes encontramos:

- Consolidación de datos relevantes de múltiples fuentes, incluyendo Big Data

- Posibilidad de elegir el método más relevante para el cliente
- Acceso a Big Data sin modelado de datos complejo
- Posibilidad de explorar asociaciones entre datos Big Data y tradicionales de base de datos
- Gráficos interactivos con Big Data
- Acceso a grandes volúmenes de datos desde dispositivos móviles
- Colaboración entre usuarios en tiempo real

Uno de los complementos más atractivos de esta herramienta es *QlinkView[18]*, integrada dentro del propio software. El análisis de Big Data con este sistema permite cargar grandes volúmenes de datos con una memoria RAM prácticamente nula. Estamos hablando de cerca de 2TB sin comprimir perfectamente manejables en un entorno de 256MB de memoria RAM, lo cual supone un gran avance cuando sabemos que cualquier ordenador de hoy en día supera por 10 a ese valor de memoria.

Es posible mediante la tecnología patentada de *Qlink*, que permite la carga binaria de documentos para acelerar la exploración de grandes volúmenes de datos. Es por ello que gran parte de su clientela está satisfecha cuando se pretenden analizar terabytes de datos almacenados en diferentes clusters Hadoop.

El enfoque hibrido entre consultas tradicionales y Big Data otorga a esta herramienta un plus sobre otras, por ejemplo:

- Posibilidad de consultar los datos en repositorios de grandes volúmenes sobre la marcha
- Recuperación de datos casi instantánea gracias al resguardo de resultados en memoria cache
- Mantiene asociaciones entre datos, independientemente de donde se encuentren estos
- Aprovechamiento de grandes volúmenes de datos sin necesidad de conocimientos previos en programación

2.2.4 Datameer

Otra de las herramientas para la integración de grandes volúmenes de datos y que se centra en el terreno del procesamiento de los mismos es *Datameer*. No es una solución opensource pero dentro del sector es bastante completa y cumple con su cometido.

Datameer simplifica el costoso análisis de datos en una única aplicación [19], apoyándose en la plataforma Hadoop. Combina el auto-servicio de datos, análisis y funcionalidad en la visualización de datos en tiempo real, de manera que resulte lo más cómodo para el usuario final.

Ofrece un análisis completo de datos, pero también un análisis predictivo complejo de datos de autoservicio. La visualización de los datos intenta desprenderse de lo simple, otorgando al

cliente unos gráficos de datos bastante vistosos y útiles en cuadros de mandos o infografías aparentemente atractivas.

Como en otro tipo de herramientas Big Data el tiempo es primordial, por lo que Datameer no se queda atrás tampoco en este aspecto. La tecnología *Smart Ejecution* patentada permite combinar y seleccionar de forma dinámica marcos de cálculo para grandes cargas de trabajo, lo que se traduce a una mayor agilidad a la hora de mostrar la información por pantalla y otorgando un rendimiento eficaz para cada proceso de análisis que ejecute el cliente. Está diseñado especialmente para aquellos casos de uso que requieren tiempos de ciclo más rápidos o con baja latencia predecible. Entre sus cualidades está el *clustering*, arboles de decisión, recomendaciones o relaciones ocultas entre datos.

Se puede destacar lo siguiente:

- Autenticación asegurada de los datos en entornos Hadoop
- Encriptación con HTTPS
- Puntos de integración flexibles para gestionar la dependencias de flujos de datos dentro de la organización
- Complejidad reducida en los análisis de datos
- Tecnología *Smart Analytics*
- Visualización de datos en tiempo real

Ilustración 15: Arquitectura Datameer 5.0[20]

2.3 Resumen general tecnologías actuales

Una vez se han expuesto las principales herramientas que se utilizan en Big Data para el tratamiento de información se puede decir que en el ámbito comercial son bastante completas en casi todos los aspectos referentes al trabajo de grandes cantidades de información.

Se ha querido enfocar las aplicaciones Big Data con tecnología Hadoop ya que la implementación de la aplicación se basa también en esa misma.

Uno de los principales inconvenientes de estas herramientas analizadas es su difícil manejo a primera instancia. Será necesario una base previa por parte del usuario para entender todo lo que proporciona cada una de las diferentes herramientas.

La gran mayoría utiliza diseños vistosos para mostrar al usuario un interfaz cómodo y atractivo, diferentes tratamientos de la información según se requiera, manejo de ficheros en HDFS de manera cómoda o incluso consolas de navegación para introducir manualmente el código que se desee. Obviamente estamos hablando de grandes compañías detrás de estas herramientas que otorgan un plus para el diseño y funcionamiento de las mismas.

No se quiere competir con ello, simplemente desarrollar una aplicación sencilla para tratar grandes volúmenes de información sin estructurar y que además permita su tratamiento mediante el paradigma MapReduce, cosa que se echa en falta en casi todas las herramientas analizadas. Por lo tanto, se puede decir que además de cumplir la finalidad con la cual se ha diseñado también permite tratar los la información eligiendo valores en MapReduce.

3 Gestión de proyecto software

El objetivo es realizar una simulación de la gestión del proyecto software desarrollado. La gestión simulará un proyecto real, realizado con las condiciones habituales del entorno empresarial. El objetivo del capítulo es plasmar los conocimientos adquiridos a lo largo de la titulación dentro de la gestión de proyectos y aplicarlos a éste como si fuera un problema real.

3.1 Alcance del proyecto

En el anterior capítulo se analizaba las diferentes tecnologías que pueden tener relación con el proyecto que aquí se expone. Una vez se ha realizado este estudio previo, comenzaremos con una breve explicación de la tecnología utilizada para el desarrollo del proyecto para así ponernos en situación.

Aunque ya se ha comentado con anterioridad, el proyecto consiste en desarrollar una herramienta que permite el manejo de ficheros HDFS mediante la plataforma Hadoop y lanzar consultas MapReduce.

Dado que el objetivo es trabajar con grandes volúmenes de datos, es necesario exponer con claridad cómo trabaja la tecnología Hadoop y cuáles son sus principales argumentos por los cuales nos hemos decantado por ello.

Con todo ello, se mostrará como la integración de Hadoop con su paradigma MapReduce pueden realizar labores casi instantáneas de consultas en bases de datos No-SQL.

3.2 Plan de trabajo

3.2...1 Identificación de tareas

Para organizar el proyecto se ha decidido identificar cada una de las tareas desarrolladas en la implementación de la herramienta. Cada tarea identifica una actividad realizada que puede ser tanto de desarrollo como de investigación o incluso documentación.

A continuación se expone la descomposición de tareas:

3.2...1.1 Estudio inicial del problema y estado del arte

Previo paso al arranque de la herramienta, debe haber un estudio previo sobre el problema que se intenta resolver, por lo que será necesario una análisis previo de la situación actual, una investigación sobre la tecnología actual que pueda guardar similitud en nuestro proyecto y que se pueda apoyar en el contexto actual.

- **Análisis del problema a tratar:** En este apartado se debe analizar o determinar cuáles son los objetivos del proyecto a desarrollar. Se debe tener en cuenta un estudio de la viabilidad en cuanto a objetivos, ya que hay que ser realistas y éstos deben estar al alcance de uno mismo.
  Dentro del entorno Hadoop no existe una herramienta que permita visualizar el contenido del fichero a tratar y los campos que contiene, para así poder facilitar la labor al realizar tareas MapReduce. Por lo tanto, se deberá realizar un estudio del problema partiendo de esa base.

- **Estado del arte:** Este apartado debe realizar un estudio previo a las tecnologías que se están utilizando en la actualidad y que tengan como contexto nuestra propia idea de la herramienta a desarrollar. Es decir, se debe analizar las distintas alternativas existentes para poder desarrollar un software que arroje algo más de iniciativa con respecto a los que existen en la actualidad para dar al consumidor algo más de lo que tiene ahora. Es por ello que este apartado es muy importante de cara a proyecto propio y con mejoras si lo comparamos con lo que hay en la actualidad. Big Data, MapReduce, Apache Hadoop, exploración de ficheros en HDFS o arquitectura Hadoop pueden ser palabras clave para comenzar con un análisis inicial que nos permita adentrarnos en el mundo de esta tecnología.

- **Conceptos adquiridos y pruebas:** Cuando se conocen las tecnologías que se aplican en desarrollos actuales sobre el tema a tratar, debemos realizar pruebas de conceptos utilizando herramientas similares a lo que nosotros mismos queremos desarrollar, para así poder tener una idea clara de lo que se permite hoy en día y hasta dónde puede llegar nuestro software. Con ello se intenta mejorar los primeros conceptos que puedan tenerse sobre el diseño, arquitectura o funcionamiento para así mejorarlos con lo que existe en la actualidad.

  Los primeros pasos en Hadoop son bastante complicados, pero existen diferentes trabajos realizados para entender el concepto de MapReduce, como puede ser el de una sonda espacial, un diccionario de 5 lenguas o un contador de palabras. Todos ellos tienen como finalidad dar a conocer el concepto del paradigma para luego uno mismo desarrollar mejores herramientas con un fin concreto.

3.2…1.2 Configuración Hadoop

El primer paso para el desarrollo del proyecto es disponer de un sistema operativo apto para integrar Hadoop [21]. Apache Hadoop es un framework que está desarrollado íntegramente en Java y además tiene licencia libre para el desarrollo de aplicaciones distribuidas, por lo que teóricamente debería estar disponible y perfectamente operativo para cualquier S.O actual. Desgraciadamente esto no es así, por lo que hay que tener en cuenta que no todos los S.O funcionen correctamente con el paradigma de MapReduce.

No existe una tabla como tal para determinar el S.O idóneo para ello, pero las diferentes pruebas realizadas por empresas y usuarios determinan cuáles son los mejores, o por lo menos los que no tienen problemas para su desarrollo. Es por ello que se debe disponer de una serie de paquetes, librerías y software específico para poder arrancar la herramienta de manera óptima y que no nos dé quebraderos de cabeza intentando determinar cuáles son las posibles causas del mal funcionamiento del paradigma.

Para el desarrollo del proyecto se ha optado por una integración de Hadoop sobre el sistema operativo ***Windows Server 2008 R2***, el cual tiene prácticamente todos el software necesario para su correcto funcionamiento. Pero para llegar hasta este punto, se han tenido que realizar diferentes pruebas de integración hasta dar con una plataforma correcta.

3.2…1.3 Selección de ficheros y limpieza de los mismos

El siguiente punto a tratar será la elección de ficheros que podrán ser leídos en la herramienta desarrollada. Como se ha comentado anteriormente, sólo nos centraremos en ficheros de texto con formato .csv y con una serie de características que deben cumplir para que la lectura del fichero sea del todo correcta.

- **Formato del fichero**
  - Debe disponer de una cabecera con todos los campos a tratar identificados por el separador ";".
  - La separación de datos debe cumplir el estándar de punto y coma, es decir, cada dato en el fichero debe venir precedido de un carácter ";" para que la lectura sea correcta.
  - El tamaño de datos en el fichero no está delimitado en ningún momento, pero se recomienda que no superen los 20Mb ya que es posible que la lectura del mismo sea extremadamente lenta y perjudique en el rendimiento de todo el programa.

Como se aprecia, el fichero de texto no tiene una gran cantidad de restricciones para su lectura, punto importante a la hora de desarrollar un programa para el cliente. Se ha tenido en cuenta este apartado para dar con la mayor amplitud a la hora de poder elegir el fichero de texto.

Todas estas restricciones se han llevado a cabo analizando multitud de ficheros .csv, su estructura y el tamaño del mismo. Por ello, se ha llegado a esta conclusión y también se entiende que puede repercutir a la hora de decantarse por este proyecto pero creemos que es la mejor forma de delimitar la lectura de ficheros a un único formato estándar.

Debido al tiempo previsto para la realización de este proyecto, se ha creído conveniente no abarcar todos los diferentes formatos de texto, ya que son muy extensos y cuyo formato puede diferir entre ellos, lo cual aumenta aún más en los diferentes tipos de ficheros de texto que existen en la actualidad.

3.2…1.4 Manejo de ficheros en Hadoop y Windows

Dado que la integración entre Windows y Hadoop debe ser completa, se debe desarrollar una herramienta que nos permita interactuar entre diferentes plataformas con los ficheros de texto. Esta herramienta debe relacionar el formato de sistema de ficheros Windows con el de Hadoop.

- **HDFS y paquetes de desarrollo Java:** Si recordamos, Hadoop dispone de su propio sistema de ficheros denominad HDFS, el cual lleva una estructura fija y que nuestro fichero de texto debe amoldar cuando tengamos intención de interactuar con el mismo dentro Hadoop. Es por eso que nuestro programa debe facilitar esta interacción, de manera que podamos cargar, borrar y modificar la ubicación del fichero que tengamos en Windows para que Hadoop pueda interpretarlo. Una vez se poseen este funcionamiento sobre Hadoop ya podremos visualizar el paradigma como resultado de la elección tomada.
  Para desarrollar esta tecnología, Hadoop dispone de paquetes de interacción con Java para así poder desarrollar el proyecto de una manera mucho más cómoda para el programador. Por poner un ejemplo, existen algoritmos que directamente permiten cargar el fichero Windows a una ubicación específica de Hadoop que nosotros mismos queramos, por lo que la tarea no debería ser muy compleja a la hora de implementar el código.

Otro punto a favor de Hadoop es que dispone de “vistas” expresamente desarrolladas para interactuar entre Eclipse y una máquina virtual de Hadoop [22]. Aunque su diseño y manejabilidad están bastante conseguidos a medida que aumentan sus versiones, no se cree conveniente utilizar este tipo de framework ya que el trabajo a desarrollar distaría mucho de una implementación pura por lo que se descartó desde el primer momento.

Por lo tanto, la interacción entre datos de diferentes sistemas de ficheros se ha realizado íntegramente en Java con la plataforma Eclipse, junto con la ayuda de paquetes de librerías desarrolladas expresamente para Hadoop.

3.2…1.5 Lectura de ficheros

Este apartado no es de los más importantes de cara al proyecto, pero se ha creído conveniente implementar una funcionalidad que permita visualizar los ficheros que estamos cargando y que obviamente se van a utilizar en el paradigma MapReduce. Es importante aclarar que la herramienta no tiene el fin de lectura de un fichero, sino las posibles relaciones entre datos de diferentes ficheros que una persona no habría podido visualizar en un primer momento.

Es decir, el paradigma MapReduce realiza un listado clave-valor que permite relacionar diferentes campos del fichero que se esté tratando, simplemente con la idea de sacar conclusiones que antes no habríamos sido capaces de reconocer a simple vista.

La lectura de ficheros es un apartado que intenta reforzar la interacción entre usuario y Hadoop, de manera que este se sienta cómodo y pueda revisar el contenido del fichero antes de lanzar el paradigma.

La idea es clara, mostrar al usuario los campos por los cuales se estructura el fichero, previsualización de todos los valores hasta un límite de 100 líneas o directamente previsualizar el contenido total del mismo si fuese necesario.

3.2…1.6 Tarea MapReduce

El punto fuerte de la aplicación es el paradigma MapReduce. Todo lo mencionado anteriormente sirve para que el funcionamiento del algoritmo sea óptimo y no repercuta en la salida generada.

Es obvio que el fichero previamente debe cumplir con los requisitos mencionados y más aún disponer del software específico para la correcta lectura en HDFS.

Se entiende que en este punto ya se conoce este paradigma y cómo actúa, pero no está mal recordarlo por si nos perdemos:

MapReduce tiene dos fases diferenciadas, *Mapper* y *Reducer [23]*. El primero de ellos es el que asigna el listado clave-valor de los campos que seleccionemos, mientras que *Reducer* se encargará de juntar todas aquellas líneas de la lista que cumplan la condición para así no disponer de líneas duplicadas con el mismo clave-valor.

Una vez se entiende el paradigma, llegamos a la fase de construcción del mismo sobre Java. Uno de los paquetes de Hadoop para Java dispone de los métodos necesarios para lanzar el paradigma, pero debe disponer de la estructura concreta ya que es muy posible que los resultados obtenidos no tengan nada que ver con lo que se esperaba. Aunque el tiempo de desarrollo se ha reducido considerablemente al tener esta herramienta, debemos recordar que la funcionalidad del mismo solo es teórica hasta este punto. Más adelante se expondrán los problemas surgidos con el mismo, ya que se ha tenido que remodelar la idea principal del algoritmo para que mostrase resultados más acordes a la finalidad del proyecto.

3.2…1.7 Evaluación del trabajo realizado

Una vez se concluye el desarrollo de la herramienta MapReduce, debemos estar convencidos que funciona perfectamente y cumple con todas las expectativas iniciales del proyecto.
Para ello no existe mejor forma que realizar multitud de pruebas unitarias sobre el sistema.

- **Evaluación del sistema de manejo de ficheros en HDFS**: Los ficheros csv deben poder leerse en su totalidad o parcialmente, dando como resultado también una serie de valores que corresponden con sus campos. Será necesario generar una serie de requisitos previos para poder comparar si realmente cumple su cometido.
  Además el usuario debe poder manejar los ficheros de su disco local con los de HDFS para tener una interacción completa.
- **Evaluación del paradigma MapReduce**: Es el corazón de la herramienta, por lo que sus pruebas deben ser exhaustivas y sobre todo cumplir las expectativas iniciales. También se dispondrán de diferentes requisitos para comparar si los cumple, multitud de archivos para realizar diferentes pruebas… todo ello con el propósito de evaluar el sistema desarrollado para dar por concluido su implementación.

3.2…1.8 Memoria TFG

Como apartado final se redactará una memoria para explicar todo el trabajo realizado hasta la fecha, indicando todos los pasos desarrollados durante los meses de implementación y búsqueda de información.
Indicar que esta memoria ha sido redactada en paralelo con la implementación del código.

3.2…2 Estimación y planificación de tareas

Para redactar este apartado se ha tenido en cuenta todas las tareas expuestas en el anterior punto. Durante todo el tiempo de desarrollo del proyecto se ha elaborado una serie de tablas para estimar la duración en horas de las diferentes etapas del proyecto.

Se ha decidido desarrollar un pequeño calendario para la estimación de tareas para luego finalizar con un diagrama de Gantt sobre esas mismas tareas planificadas.

3.2…2.1 Calendario

Para una mejor compresión de las tareas realizadas y del tiempo fijado para cada una de ellas, se adjunta una tabla donde se desglosa el tiempo entre todos los hitos a desarrollar y que se han explicado con anterioridad en el apartado previo.

TAREA	FECHA INICIO	FECHA FIN	DURACIÓN HORAS	DURACION DIAS
1. Estudio inicial del problema y estado del arte	15/08/14	27/10/14	85	103
2. Configuración Hadoop	04/10/14	21/10/14	35	17
3. Selección de ficheros y limpieza de los mismos	26/10/14	11/11/14	29	16
4. Manejo de ficheros en Hadoop y Windows	02/11/14	09/11/14	15	7
5. Lectura de ficheros	15/11/14	24/12/14	36	39
6. Tarea MapReduce	18/11/14	29/12/14	50	41
7. Evaluación del trabajo realizado	12/12/14	16/01/15	74	34
8. Memoria TFG	19/09/14	26/01/15	95	129
TOTAL	15/08/14	19/02/15	419	174

Tabla 1: Calendario de tareas

Como se aprecia, el proyecto comenzó el día 15 de Agosto del 2014 y finalizó el día 19 de Febrero del 2015, con un total de 560 días aproximadamente. Hay que tener en cuenta que algunos festivos y fines de semana también se han contabilizado en estas fechas ya que se ha trabajado en el proyecto, pero no todos los días contabilizados se han trabajado sino que han sido parte del recorrido del desarrollo de todo el proyecto.
El total de horas empleadas para el proyecto son de 482, aproximadamente. Se están contabilizando las horas reales de desarrollo o búsqueda de información, pero no se ha realizado como un trabajo continuo durante las fechas señaladas. Es decir, algunos días no se ha realizado nada del proyecto y se contabiliza en el número de días pero no se contabiliza el número de horas de esa fecha al no tratarse de un avance en el proyecto.

Dado que cada tarea a desarrollar dispone de algunos apartados, se ha querido tener en cuenta el desglose de cada subtarea asociada. A continuación se exponen el mismo formato de tabla pero con la subtarea asociada. En el caso de no disponer de subtarea, no se ha incluido

- **Estudio inicial del problema y estado del arte**

TAREA	FECHA INICIO	FECHA FIN	DURACIÓN HORAS	DURACION DIAS
Análisis del problema a tratar	15/08/14	12/09/14	23	35
Estado del arte	15/09/14	27/10/14	47	42
Conceptos adquiridos y pruebas	15/08/14	27/10/14	15	73
TOTAL	15/08/14	27/10/14	85	103

Tabla 2: Calendario estudio inicial del problema y estado del arte

- **Selección de ficheros y limpieza de los mismos**

TAREA	FECHA INICIO	FECHA FIN	DURACIÓN HORAS	DURACION DIAS
Formato del fichero	26/10/14	11/11/14	29	16

Tabla 3: Calendario selección de ficheros y limpieza de los mismos

- **Manejo de ficheros en Hadoop y Windows**

TAREA	FECHA INICIO	FECHA FIN	DURACIÓN HORAS	DURACION DIAS
HDFS y paquetes de desarrollo Java	02/11/14	09/11/14	15	7

Tabla 4: Calendario manejo de ficheros en Hadoop y Windows

- **Evaluación del trabajo realizado**

TAREA	FECHA INICIO	FECHA FIN	DURACIÓN HORAS	DURACION DIAS
Evaluación del sistema de lectura de csv	12/12/14	16/01/14	25	34
Evaluación del paradigma MapReduce	12/12/14	16/01/14	49	34

Tabla 5: Calendario evaluación del trabajo realizado

3.2…2.2 Diagrama de Gantt

El Diagrama de Gantt[24] es una herramienta bastante útil cuyo objetivo no es más que exponer el tiempo dedicado a una tarea en cuestión, el cual puede mostrar de forma gráfica una evolución de los días consumidos por cada tarea para así poder tener una idea más clara del posible tiempo consumido en una tarea.

Esta herramienta es utilizada en cualquier proyecto para realizar una representación gráfica del progreso del mismo además de ser un buen medio de comunicación entre diversas personas que estén involucradas en el proyecto. Desgraciadamente esta herramienta no permite ver las posibles relaciones entre actividades. A pesar de ello, se ha realizado un diagrama de Gantt donde se especifican todas las tareas mencionadas con anterioridad.

Entre la información relevante que puede aportar el diagrama de Gantt se puede encontrar:

- Periodos de tiempo entre diferentes tareas
- Referencias a las subtareas con respecto a su tarea padre en conjunto
- Dependencias entre tareas

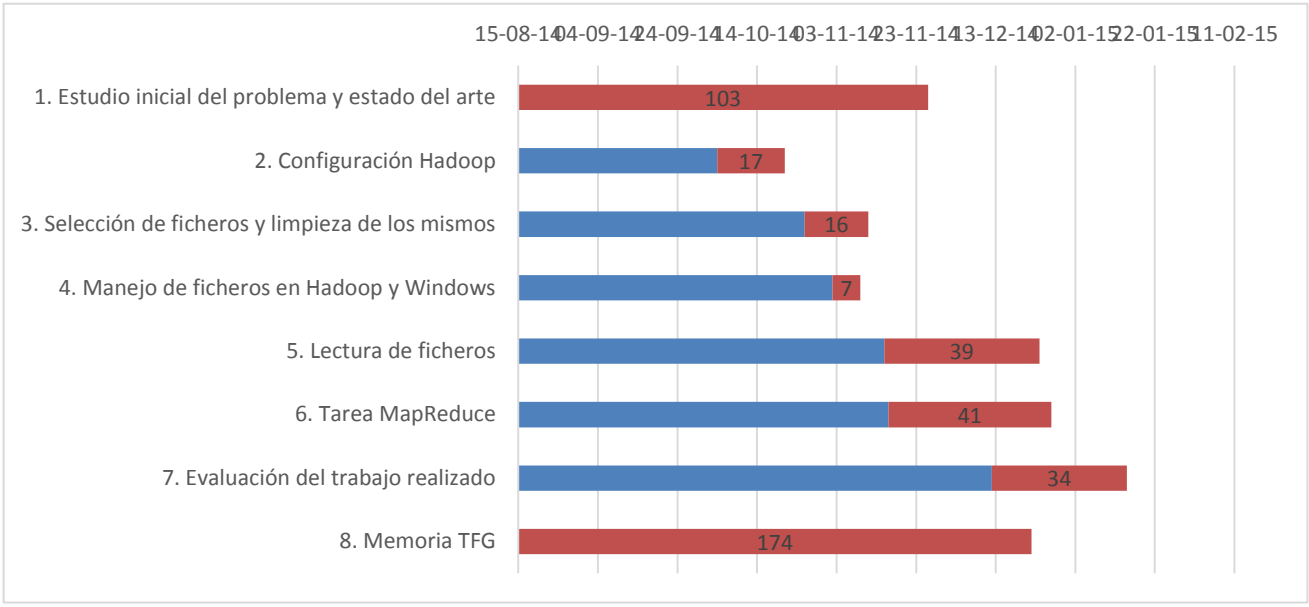


Ilustración 16: Diagrama de Gantt

El diagrama de Gantt expuesto en la figura anterior representa las tareas realizadas a lo largo del desarrollo del proyecto, con fecha de inicio consta a día 15-08-2014.

Las tareas iniciales son la numero 1 y la 8 que empiezan al arrancar el proyecto mientras que las restantes van empezando según se finalizan otras, ya que están ligadas entre ellas y sería imposible comenzar una sin concluir la anterior. No en todos los casos se da, pero la mayoría sí que necesita finalizar la anterior etapa para comenzar una nueva.

Se ha resaltado el número de días de cada tarea donde se puede apreciar que el máximo tiempo invertido ha sido en la memoria del TFG, unas 174 horas en total. Es normal, ya que cada nueva tarea requiere incluir detalles en la memoria aunque no se gestione el total de la misma hasta que no se finalice la tarea. Es decir, una vez se ha terminado la tarea se plasmará el resultado en la memoria.

También se puede observar que la tarea de recopilación de información y el estudio inicial del problema requiere un tiempo bastante importante en la planificación del proyecto. Es necesario que existe un mínimo de tiempo para comprobar el problema y como llevarlo a cabo, mientras que el estudio de aplicaciones o software dentro del mundo Big Data requiere conocerse al detalle con el fin de obtener una aplicación que pueda mejorar a las ya existentes.

Destaca también que las tareas de desarrollo de la aplicación, número 5 y 6, también requieren cierto tiempo en su totalidad. Es consecuente, ya que la implementación del código para la aplicación es costosa en tiempo y pueden surgir percances que en un primer momento no se han tenido en cuenta.

La planificación inicial en tiempo no se realizó en primera instancia ya que no se creía conveniente indicar fechas de finalización cuando no se disponía claramente del problema a tratar y cuales podían ser las soluciones para llevarlas a cabo. Por lo tanto, este diagrama de Gantt solo expone las fechas finales de duración de las tareas realizadas pero con carácter real.

3.3 Gestión de recursos

Este apartado sirve como referencia para detallar el presupuesto del proyecto “Interfaz de consultas MapReduce para Hadoop”. Para realizar tal tarea se ha tenido en cuenta los costes materiales del proyecto al igual que los costes personales, todo ello siguiendo una plantilla presupuestaria que facilita la Universidad Carlos III de Madrid.

No se ha tenido en cuenta los costes en dietas y viajes, al igual que tampoco se añaden los costes de materiales fungibles, como puede ser la luz, impresiones realizadas…

Para costear el presupuesto debe tenerse en cuenta aspectos como:

- Los costes serán expresados en su totalidad por euros
- Se tomarán dos decimales para cantidades económicas y si es necesario, redondear.

Para finalizar el cálculo se incluyen las herramientas utilizadas para el desarrollo del proyecto, como pueden ser los equipos utilizados.

El fin de indicar los presupuestos es realizar un balance del tiempo de desarrollo del proyecto y del coste que tendría para una empresa real.

3.3…1 Personas involucradas en el proyecto

Durante todo el proceso del desarrollo del proyecto solo han participado tres personas:

- **Jorge Rivert González:** Con categoría de Ingeniero Junior, es la persona encargada de todo el desarrollo del proyecto, desde el análisis de las tecnologías hasta la implementación total de la herramienta.
  Dado que fue desarrollado por esta persona, se ha tenido que adjudicar el rol de autor del proyecto.

- **Harith Taha Abdulla Aljumaily:** Con categoría de Ingeniero y rol de tutor del proyecto, es la persona encargada de dirigir la evolución del mismo dando ideas y mejoras para el desarrollo de la herramienta.

- **María Dolores Cuadra Fernández**: Con categoría de Ingeniero y rol de Directora del proyecto, es la persona encargada de coordinar a las dos anteriores en tareas referentes a la evolución del proyecto, posibles mejoras a implementar y análisis de la herramienta final para ver si cumple con su cometido.

Una vez disponemos de las personas encargadas del proyecto se mostrará en una tabla a cada una de ellas para desglosar la categoría, el coste por horas, horas totales invertidas en el proyecto y finalmente el coste total.

NOMBRE	CATEGORÍA	COSTE HORA	HORAS TOTALES	COSTE TOTAL
Jorge Rivert González	Ingeniero Junior	11	419	5.786 €
Harith Taha Abdulla Aljumaily	Ingeniero	21	19	399 €
María Dolores Cuadra Fernández	Ingeniero	26	8	208 €
TOTAL				6.393 €

Tabla 6: Personal encargado del proyecto

3.3…2 Costes de los elementos software y hardware

Este apartado indica los equipos utilizados para el desarrollo del proyecto, componentes hardware y software que han sido necesarios para concluir el TFG de manera óptima y rápida.

A continuación se expone una tabla que muestra los componentes y su precio.

DESCRIPCION	TIEMPO DEDICADO (MESES)	UTILIZACION EN %	PRECIO
Ordenador portátil Asus N55SF	4	100	987 €
Disco duro Seagate 2TB	2	30	115 €
Windows Server 2008 R2	4	100	623,86 €
Hadoop	2	75	Gratuito
Eclipse	4	100	Gratuito
TOTAL			1.725,86 €

Tabla 7: Componentes utilizados en el proyecto

3.3…3 Coste total del proyecto

Si tenemos en cuenta la tabla de costes personales y la de materiales utilizados, podremos sacar el coste total del proyecto, es decir, el aporte total que debe realizarse para llevarse a cabo por una empresa

DESCRIPCIÓN	PRECIO
Coste del personal	6.393 €
Coste materiales	1.725,86 €
TOTAL	8.118,86 €

Tabla 8: Coste total del proyecto

Por lo tanto, el proyecto tiene como coste final un coste de *ocho mil ciento dieciocho con ochenta y seis euros*

Ahora que disponemos del coste final del proyecto, debemos ajustarlo para obtener unos beneficios del mismo. Es por ello que la siguiente tabla incluye un beneficio del 18% sobre el total del coste del proyecto, por lo que quedaría de la siguiente manera:

DESCRIPCIÓN	PRECIO
Coste del personal	6.393 €
Coste materiales	1.725,86 €
Beneficios (18%)	1.461,39 €
TOTAL	9580.25 €

Tabla 9: Coste total del proyecto con beneficios

Ahora esta tabla muestra la totalidad del presupuesto del proyecto que asciende *a nueve mil quinientos ochenta con veinticinco euros (9580,25 €),* de los cuales 1.461,39 € son en concepto de beneficios.

3.4 Gestión de riesgos

3.4…1 Riesgos

Este apartado trata la incertidumbre relativa a una amenaza que pueda existir en el ciclo de vida del proyecto. Se debe realizar una secuencia de actividades como evaluar el riesgo, utilizar diferentes estrategias de desarrollo para controlarlo o mitigar el riesgo producido utilizando recursos gerenciales. Es resumidas cuentas, se intenta evadir el riesgo, reducir sus efectos negativos o aceptar un riesgo en particular.

3.4…2 Identificación y Análisis de riesgos

INFORME DE RIESGO		
Tipo de incidencia	Técnica	**Impacto ALTO**
Causa	**Requisitos no detallados**	
Descripción breve de la solución	Los requisitos son parte esencial en el análisis de proyecto, pero es posible que puedan variar a lo largo del ciclo de vida. Por lo tanto, los requisitos poco detallados tendrán problema a la hora de ponerlos en práctica y se recomienda especificarlos con gran detalle	

Tabla 10: Informe de riesgo 1

INFORME DE RIESGO		
Tipo de incidencia	Técnica	**Impacto ALTO**
Causa	**Complejidad interfaces**	
Descripción breve de la solución	Se debe identificar como actuarán las interfaces entre sí	

Tabla 11: Informe de riesgo 2

INFORME DE RIESGO		
Tipo de incidencia	Técnica	**Impacto MEDIO**
Causa	**Rendimiento y fiabilidad**	
Descripción breve de la solución	Dado que es un proyecto nuevo y no comparable en su totalidad con otras aplicaciones existentes, es imposible estimar la fiabilidad del sistema y la velocidad del mismo	

Tabla 12: Informe de riesgo 3

INFORME DE RIESGO		
Tipo de incidencia	Organizativo	**Impacto MEDIO**
Causa	**Dependencias**	
Descripción breve de la solución	La aplicación se estructura en varios interfaces que deben integrarse para formar uno mismo dentro de un interfaz principal. Es posible que las dependencias existentes entre diferentes interfaces puedan obstaculizar al conjunto final.	

Tabla 13: Informe de riesgo 4

INFORME DE RIESGO		
Tipo de incidencia	Técnica	**Impacto**
Causa	**Requisitos no detallados**	
Descripción breve de la solución	Los requisitos son parte esencial en el análisis de proyecto, pero es posible que puedan variar a lo largo del ciclo de vida. Por lo tanto, los requisitos poco detallados tendrán problema a la hora de ponerlos en práctica y se recomienda especificarlos con gran detalle	

Tabla 14: Informe de riesgo 5

INFORME DE RIESGO		
Tipo de incidencia	Organizativos	**Impacto MEDIO**
Causa	**Financiación**	
Descripción breve de la solución	El presupuesto inicial puede verse afectado por la coyuntura económica	

Tabla 15: Informe de riesgo 6

INFORME DE RIESGO		
Tipo de incidencia	Organizativos	**Impacto BAJO**
Causa	**Recursos y prioridad**	
Descripción breve de la solución	La falta de recursos para el desarrollo del proyecto pueden ser un obstáculo para cumplir los plazos de entrega	

Tabla 16: Informe de riesgo 7

INFORME DE RIESGO		
Tipo de incidencia	Gestión del proyecto	**Impacto ALTO**
Causa	**Planificación**	
Descripción breve de la solución	No se dispone de una planificación total que permita saber si los plazos del desarrollo se están cumpliendo. Será necesario disponer de una tabla de planificación para reducir en gran medida los riesgos existentes en este aspecto.	

Tabla 17: Informe de riesgo 8

INFORME DE RIESGO		
Tipo de incidencia	Gestión del proyecto	**Impacto MEDIO**
Causa	**Control**	
Descripción breve de la solución	Se debe controlar todos los cambios que puedan surgir a lo largo del proyecto en una tabla de control de cambios para poder valorar el progreso del mismo.	

Tabla 18: Informe de riesgo 9

INFORME DE RIESGO		
Tipo de incidencia	Gestión del proyecto	**Impacto MEDIO**
Causa	**Comunicación**	
Descripción breve de la solución	Pueden existir informes poco aclaratorios sobre la evolución del proyecto	

Tabla 19: Informe de riesgo 10

INFORME DE RIESGO		
Tipo de incidencia	Técnica	**Impacto ALTO**
Causa	**Estimación**	
Descripción breve de la solución	La estimación del trabajo puede ser incompleta o parcial	

Tabla 20: Informe de riesgo 11

3.5 Plan de pruebas

Este apartado tiene como finalidad especificar las pruebas que van a ser realizadas para comprobar la funcionalidad de la aplicación para observar si realmente cumple con las tareas asignadas que se han explicado con anterioridad.

Hay que tener en cuenta que las pruebas se irán desarrollando una vez se avance en la implementación de la herramienta, por lo que muchas de las pruebas no se podrán analizar hasta finalizar con la totalidad del desarrollo del proyecto.

3.5…1 Definición y especificación de pruebas

Dado que solo se dispone de una herramienta implementada, se ha querido diferenciar las diferentes pruebas realizadas con una división de manejabilidad del software. Por ello, las pruebas estarán divididas en tres categorías:

- Pruebas navegación principal y manejo de ficheros Hadoop
- Pruebas de Explorador de ficheros
- Pruebas MapReduce

Para cada una de las divisiones realizadas se han seguido estas pruebas [25]:

- **Pruebas unitarias:** Si tenemos en cuenta que el proyecto se basa en una programación orientada a objetos, las pruebas unitarias se ha realizado sobre cada método realizado siempre que estas tengan sentido en el contexto. Estas pruebas intentan reflejar el correcto funcionamiento de los métodos implementados para cada una de las entradas posibles que se le pueda dar y siempre verificando que la salida obtenida se corresponda con el resultado correcto que se espera.
  Todas estas pruebas han sido realizadas con un framework específico del lenguaje Java: JUnit

- **Pruebas funcionales:** Este apartado de pruebas se realizan una vez se ha finalizado con el anterior banco de pruebas. Este tipo de pruebas hacen referencia al funcionamiento completo de la aplicación, por lo que se debe verificar en su totalidad. Es importante resaltar que estas pruebas están realizadas con parte del contenido de entrada que se ha verificado para las pruebas unitarias, es decir, se han escogido entradas válidas y funcionales para verificar la integridad total de la herramienta.

- **Pruebas de rendimiento:** Cuando ya conocemos el funcionamiento total de la aplicación mediante el banco de pruebas anteriores, es la hora de verificar que la aplicación puede ejecutarse sobre diferentes condiciones de trabajo para así poder valorar la eficiencia del mismo. Este tipo de pruebas están orientadas a

reflejar tiempos de ejecución sobre distintos recursos para evaluar la escalabilidad de la herramienta.
En nuestro caso se ha forzado al sistema de ficheros mediante carga computacional alta. Esto es posible gracias a ficheros de texto cuyo tamaño supere los 5mb, ya que se ha comprobado que el paradigma MapReduce necesita un tiempo considerable debido a la cantidad de procesamiento de valores, al igual que la previsualización del fichero.

Para una mejor comprensión por parte del lector, se detallarán todas las pruebas realizadas en una tabla donde se especifica la finalidad de la prueba, los pasos que se han seguido para llevar a cabo esa prueba y el resultado obtenido.

A continuación se deja una leyenda para entender la tabla:

IDENTIFICADOR	FINALIDAD	PASOS	FECHA PRUEBA	RESULTADO
Prueba. X-Y				

Tabla 21: Plantilla pruebas

Dónde:

- **Identificador**: Es único y sirve para identificar el número de prueba (X) y las subtarea (Y).
- **Finalidad**: Es una descripción breve de la prueba realizada al igual que se especifica cual es el propósito de la prueba en sí.
- **Pasos**: Se indica el proceso para llevar a cabo la prueba a realizar
- **Fecha prueba**: Indica la fecha en la cual se ha realizado la prueba
- **Resultado**: Puede ser satisfactorio o producirse un error en el desarrollo de la prueba

Pruebas navegación principal y manejo de ficheros Hadoop

ID	FINALIDAD	PASOS	FECHA PRUEBA	RESULTADO
Prueba. 1-1	Arrancar la aplicación y mostrar el entorno visual del proyecto	1. Arrancar la aplicación	11-01-2014	Éxito
Prueba. 1-2	Arrancar la aplicación y mostrar el listado HDFS	1. Arrancar la aplicación	11-01-2014	Éxito
Prueba. 1-3	Seleccionar un fichero HDFS del listado izquierdo	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop	11-01-2014	Éxito
Prueba. 1-4	Borrar un fichero cargado en el directorio de Hadoop mediante el botón de pantalla principal	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Clicar sobre el botón “delete”	11-01-2014	Éxito
Prueba. 1-5	Refrescar listado principal de ficheros Hadoop	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Clicar sobre el botón “Refresh”	11-01-2014	Éxito
Prueba. 1-6	Listar ficheros Hadoop con sus campos sobre la pantalla principal. Debe aparecer una caja de texto con los campos en los cuales se distribuye el fichero .csv	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Doble click sobre el fichero	11-01-2014	Éxito
Prueba. 1-7	Comprobar dimensionalidad de las cajas de texto mediante la propiedad que permite ajustar el tamaño al que uno desee	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Doble click sobre el fichero 4. Reajustar dimensionalidad de la caja	11-01-2014	Éxito

Prueba. 1-8	Minimizar cajas de texto mediante el botón de la barra de tareas de la propia caja	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Doble click sobre el fichero 4. Click sobre minimizar en la caja de texto	11-01-2014	Éxito
Prueba. 1-9	Cerrar la caja de texto mediante el botón de la barra de tareas de la propia caja	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Doble click sobre el fichero 4. Click sobre cerrar en la caja de texto	11-01-2014	Éxito
Prueba. 1-10	Visualización de múltiples cajas de texto para comprobar cantidad de cajas en pantalla	1. Arrancar la aplicación 2. Seleccionar fichero en ruta Hadoop 3. Doble click sobre el fichero múltiples veces	11-01-2014	Éxito
Prueba. 1-11	Comprobar funcionalidad de la aplicación cuando no se arranca el clúster Hadoop. Debe apreciarse un error en la consola Java donde se especifica que no se tiene acceso al clúster	1. Arrancar la aplicación 2. Deshabilitar el arranque del clúster Hadoop	11-01-2014	Éxito

Tabla 22: Pruebas navegación principal y manejo de ficheros Hadoop

Pruebas de Explorador de ficheros:

Este tipo de pruebas se basan en el entorno exclusivo para manejo de ficheros de Windows junto con Hadoop, mediante el formulario de exploración de ficheros. Se entiende que el arranque del programa ya se ha llevado a cabo, por lo que ese proceso se dejará al margen en la explicación de la tabla.

ID	FINALIDAD	PASOS	FECHA PRUEBA	RESULTADO
Prueba. 2-1	Arranque del explorador de ficheros una vez estemos en el formulario principal, mediante el botón de "File" que se encuentra en la barra superior de la herramienta	1. Arrancar el explorador de ficheros mediante el botón "File" de la barra superior del navegador	11-01-2014	Éxito
Prueba. 2-2	Visualización del árbol de ficheros propio de nuestro sistema operativo, el cual reflejará todo el árbol de ficheros y carpetas de nuestro HDD	1. Arrancar el explorador de ficheros	11-01-2014	Éxito
Prueba. 2-3	Selección de ficheros	1. Arrancar el explorador de ficheros 2. Seleccionar fichero del árbol de Windows	11-01-2014	Éxito
Prueba. 2-4	Concordancia de manejo de botones una vez se elige un fichero de la navegación. El resultado tiene que otorgar visibilidad de botones propios del fichero	1. Arrancar el explorador de ficheros 2. Seleccionar fichero del árbol de Windows 3. Comprobar disponibilidad de botones	11-01-2014	Éxito
Prueba. 2-5	Cargar un fichero Windows a la ruta establecida Hadoop mediante el navegador de ficheros. Una vez se seleccione el fichero y se cargue, debe estar disponible en el listado	1. Arrancar el explorador de ficheros 2. Seleccionar fichero del árbol de Windows 3. Clicar sobre el botón "Load"	11-01-2014	Éxito

	principal de la ruta Hadoop	4. Comprobar si el fichero se ha cargado correctamente en la ruta		
Prueba. 2-6	Previsualización de ficheros csv en el entorno de navegación de ficheros. El fichero debe mostrar un máximo de 100 líneas en el cuadro adjunto de la parte derecha del cuadro de navegación de Windows	1. Arrancar el explorador de ficheros 2. Seleccionar fichero del árbol de Windows 3. Clicar sobre el botón “View” 4. Comprobar que se ha cargado el contenido en la parte derecha	11-01-2014	Éxito
Prueba. 2-7	Visualización completa del fichero mediante el botón “View” y en el enlace inferior que indica “Show me more…”. El resultado se reflejará en forma de bloc de notas con el contenido total del fichero seleccionado	1. Arrancar el explorador de ficheros 2. Seleccionar fichero del árbol de Windows 3. Clicar sobre el botón “View” 4. Comprobar que se ha cargado el contenido en la parte derecha 5. Clicar sobre el botón “Show me more…” 6. Se abre un bloc de notas con el contenido total del fichero	11-01-2014	Éxito
Prueba. 2-8	Comprobar carga de ficheros .csv. Se debe comprobar si es posible cargar cualquier tipo de fichero, aunque no se pueda realizar MapReduce sobre él.	1. Arrancar el explorador de ficheros 2. Seleccionar fichero .csv del árbol de Windows 3. Clicar sobre el botón “Load”	11-01-2014	Éxito
Prueba. 2-9	Realizar varias iteraciones de manejo de ficheros para establecer si se accede al fichero correcto	1. Arrancar el explorador de ficheros 2. Seleccionar	11-01-2014	Éxito

	y no es siempre el mismo	fichero del árbol de Windows 3. Seleccionar múltiples veces las opciones de "Load" o "View" sobre distintos ficheros		
Prueba. 2-10	Cerrar el explorador de ficheros Windows	1. Arrancar el explorador de ficheros 2. Clicar sobre el botón de aspa para cerrar	11-01-2014	Éxito
Prueba. 2-11	Carga de ficheros .csv de alto contenido y tamaño para rendimiento del sistema en caso de visualización completa del fichero	1. Arrancar el explorador de ficheros 2. Cargar un fichero .csv con un peso medio de 5mb y realizar una previsualización del mismo	11-01-2014	Éxito

Tabla 23: Pruebas Explorador de ficheros

El siguiente apartado de pruebas es apto para el paradigma MapReduce, por lo que se entiende que el usuario ya tiene su listado de ficheros Hadoop cargados y listos para generar unos resultados. La finalidad radica en comprobar que se puede manejar los distintos campos con los cuales consta el fichero y además es posible elegir cualquiera de ellos para realizar el paradigma.

ID	FINALIDAD	PASOS	FECHA PRUEBA	RESULTADO
Prueba. 3-1	Arrancar el formulario de MapReduce con el botón de "MapReduce" que se encuentra en la barra de herramientas	1. Arrancar el formulario de MapReduce	11-01-2014	Éxito
Prueba. 3-2	Visualización de todos los ficheros ubicados en la carpeta HDFS de MapReduce en el listado izquierdo del formulario. Previamente se debe disponer de ficheros	1. Arrancar el formulario de MapReduce 2. Visualizar todos los ficheros que se han cargado previamente	11-01-2014	Éxito

	cargados en HDFS para poder seleccionar alguno de ellos			
Prueba. 3-3	Selección de ficheros en el listado HDFS. Mediante esta acción se debe habilitar el siguiente listado con los campos del fichero para así poder seleccionar un o varias keys	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Comprobar que se abre un nuevo listado para selección de key/s	11-01-2014	Éxito
Prueba. 3-4	Selección de key/s una vez se elige el fichero, comprobando que los campos mostrados en el listado de key/s corresponden al fichero. Cuando se seleccione un key/s, se debe abrir el siguiente listado con los campos del mismo fichero que muestran la selección de value/s	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar una key/s 4. Se abre un nuevo listado con los mismos campos que en el listado anterior pero en un nuevo formulario	11-01-2014	Éxito
Prueba. 3-5	Seleccionar un campo del listado de value/s y comprobar que se guardan los valores seleccionados previamente. Se debe desbloquear los botones de selección de uno o todos los ficheros de la carpeta HDFS, además del botón de MapReduce ubicado al final del proceso de selección	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar una key/s 4. Seleccionar un value/s 5. Comprobar que se deshabilitan los botones de uno o varios ficheros	11-01-2014	Éxito
Prueba. 3-6	Seleccionar el botón de uno o varios ficheros al final del proceso de selección. Se debe habilitar el botón MapReduce con cualquiera de las dos opciones	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar una key/s 4. Seleccionar un	11-01-2014	Éxito

		value/s 5. Seleccionar uno o todos los ficheros de la carpeta HDFS 6. El botón MapReduce ya está disponible		
Prueba. 3-7	Clicar sobre el botón "MapReduce" para comenzar con el proceso del paradigma. El resultado debe mostrar un formulario con la opción de visualizar el fichero generado por el paradigma	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar una key/s 4. Seleccionar un value/s 5. Seleccionar uno o todos los ficheros de la carpeta HDFS 6. Clicar sobre "MapReduce"	11-01-2014	Éxito
Prueba. 3-8	Visualización del fichero resultado del paradigma MapReduce y comprobar que los campos seleccionados para Key/s y value/s son los correctos, ya que se han seleccionado previamente	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar una key/s 4. Seleccionar un value/s 5. Seleccionar uno o todos los ficheros de la carpeta HDFS 6. Clicar sobre "MapReduce" 7. Clicar sobre "Yes" 8.Comprobar resultado	11-01-2014	Éxito
Prueba. 3-9	Realizar proceso de selección de ficheros para MapReduce, pero incluyendo más de una Key y un Value en la selección del listado. El resultado tiene que	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado 3. Seleccionar más de una key 4. Seleccionar más	11-01-2014	Éxito

	reflejar todos los campos que se han seleccionado y cumplir el paradigma con esos valores	de un value 5. Seleccionar uno o todos los ficheros de la carpeta HDFS 6. Clicar sobre "MapReduce" 7. Clicar sobre "Yes" 8.Comprobar resultado		
Prueba. 3-10	Cerrar el explorador de MapReduce	1. Arrancar el formulario de MapReduce 2. Clicar sobre el aspa del formulario y esperar a su cierre	11-01-2014	Éxito
Prueba. 3-11	Realizar MapReduce sobre un fichero HDFS cuyo tamaño supere los 5mb para comprobar la estabilidad del sistema y velocidad del paradigma MapReduce	1. Arrancar el formulario de MapReduce 2. Seleccionar un fichero del listado cuyo tamaño sea superior a 5mb 3. Seleccionar una key 4. Seleccionar un value 5. Seleccionar un fichero de la carpeta HDFS 6. Clicar sobre "MapReduce" 7. Clicar sobre "Yes" 8.Comprobar resultado	11-01-2014	Éxito

Tabla 24: Pruebas MapReduce

3.5…2 Evaluación final de las pruebas

Los objetivos esperados antes de analizar los resultados de las pruebas deberían reflejar éxito absoluto si tenemos en cuenta que el software debe estar pulido y perfectamente funcional, por lo que no tendría sentido tener como conclusiones al finalizar la pruebas un fracaso en alguna de ellas.

Es por ello que todas las pruebas realizadas reflejan éxito, lo cual es obligatorio para el proyecto. Si además tenemos en cuenta que las pruebas se centran más en el funcionamiento

principal de la aplicación y no tanto en la interacción con el usuario, tenemos un producto acorde a lo que especifica el cliente en un primer momento.

En cuanto a los resultados obtenidos de las pruebas, se aprecia claramente que el producto funciona correctamente en todos sus apartados. Se ha hecho más hincapié en las pruebas funcionales, como la carga de ficheros o MapReduce con múltiples valores de entrada para forzar al sistema a una carga computacional alta y demostrar así que el funcionamiento del software es muy bueno.

Desgraciadamente las pruebas, en cuanto a rendimiento se refieren, no han sido del todo satisfactorias. La carga de ficheros con un peso importante, alrededor de los 5mb, produce una saturación en el clúster de Hadoop y entorpecen el rápido procesamiento del paradigma MapReduce. Esto es debido a que se están realizando las pruebas sobre un único nodo, lo cual genera un cuello de botella cuando se deben analizar gran cantidad de datos. Normalmente este tipo de banco de pruebas debe realizarse sobre varios nodos interconectados para que la distribución de las tareas se realice en paralelo, lo cual minimiza la carga computación al ser distribuida en varios nodos.

Aunque no se haya podido comprobar este último dato, la teoría y las diferentes pruebas realizadas por otros usuarios nos indica que el procesamiento de valores por parte del paradigma es tremendamente alto, cosa que no se ha podido reflejar claramente con ficheros de tamaño considerable en nuestro sistema. Es una gran pena, ya que el potencial de MapReduce radica especialmente en ficheros con un gran tamaño (estamos hablando de gigas de información en un solo fichero) pero que debido al presupuesto y materiales disponibles no se ha podido contemplar.

Otro de los aspectos importantes de los resultados de las pruebas realizadas es la manejabilidad de ficheros del sistema Windows con HDFS. Aunque a simple vista parezca ínfimo este aspecto, debemos tener en cuenta que la carga de un sistema de ficheros como es Windows a otro como es HDFS requiere que sea casi instantáneo. Este punto se ha conseguido con creces, ya que el manejo de ficheros HDFS en Hadoop mediante la herramienta es casi como en Windows, junto con un interfaz acorde a lo que estamos acostumbrados y que ayuda bastante a familiarizarse con el entorno.

Como resultado final de las pruebas podemos concluir que han sido satisfactorias casi en su totalidad, lo que refleja un producto terminado con un funcionamiento completo.

4 Solución

Este apartado refleja la solución que se ha llevado a cabo. Por ello se debe exponer todo el proceso de desarrollo del mismo junto con las pruebas realizadas una vez se finalice la implementación del código.

4.1 Definición del problema: Soluciones

El trabajo desarrollado consiste en el diseño e implementación de un interfaz que permita realizar consultas MapReduce para Hadoop. El objetivo de este trabajo es obtener consultas sobre bases de datos NO-SQL mediante la lectura de cualquier fichero de texto que cumpla con los requisitos de un ".csv". Para ello será indispensable trabajar con un entorno Hadoop, ya que será necesario para poder realizar el paradigma MapReduce.

Este paradigma es el objetivo principal de la aplicación a desarrollar. Mediante el mismo se pueden obtener cualquier tipo de consulta relacional que se nos ocurra con un fichero o varios a la vez y en un tiempo reducido, que es uno de los puntos fuertes de Hadoop. Con ello se intenta transmitir al usuario que las bases de datos tradicionales como hoy las conocemos están en un periodo de adaptación hacia No-SQL, es decir, no será necesario dotar al fichero de una estructura predefinida para poder realizar las consultas. La idea es que el cliente genere consultas MapReduce sobre el fichero sin tener en cuenta el tipo de formato del mismo.

Esto abre muchas puertas de cara al futuro, ya que a medida que internet y sobre todo las redes sociales, aumentan día tras día, su volumen de datos también lo hace pero de manera exponencial. Este tipo de datos que ahora mismo no se están utilizando en su gran mayoría son los denominados Big Data, que poco a poco se dan a conocer. Si tenemos multitud de datos que pueden otorgarnos una información realmente útil, ¿Por qué no utilizarlos?

La tecnología ahora lo permite, pero se necesita una gran cantidad de procesamiento para consultar datos relevantes en un corto periodo de tiempo[26], ya que cada segundo se está generando información que debe ser tratada y eso requiere un alto procesamiento de información que una base de datos relacional no puede ni plantearse en la actualidad.

Por ello se ha querido desarrollar una interfaz cómoda para el usuario final de manera que este tenga un entorno familiar, como es Windows, pero que permite interaccionar con valores o datos HDFS y sacar resultados de las consultas que ellos mismos necesiten.

Por ejemplo, se dispone de nóminas de personas de una empresa y se quiere consultar cuales son las que ganan más dinero en un periodo de tiempo establecido. Para ello se necesitaría disponer de una base de datos relacional que incluyera los datos referentes a la nómina durante X tiempo, lo cual se ha debido de realizar antes de esta consulta.

Ahora imagínate que necesitamos esa consulta pero disponemos de un programa capaz de realizar una lectura de cualquier fichero de texto que venga estructurado sin la necesidad de haber generado una base de datos previa. Impensable ¿no?, pues nos equivocamos ya que la herramienta a desarrollar puede realizar esa tarea de manera sencilla y muy rápida.

4.2 El proceso de desarrollo

En este apartado se explica todo el proceso de desarrollo de la herramienta implementada, desde su fase de análisis hasta las pruebas realizadas.

Existen diferentes modelos de procesos para la Ingeniería de software. Cada uno de ellos pretende proporcionar un orden al proceso del desarrollo del software. Por ello es necesario incluir un modelo que se aproxima mejor a la organización de las actividades que se plantean en base a una serie de etapas interconectadas entre sí.

Para este proceso se seguirá el **Modelo Lineal Secuencial**, conocido como en cascada. Se basa en un enfoque sistemático y secuencial en cuanto al desarrollo del software se refiere, comenzando por un nivel de sistemas y finalizando en las pruebas realizadas sobre el software.

La ingeniería del software tiene un modelo lineal secuencial que se ilustra en la siguiente figura:

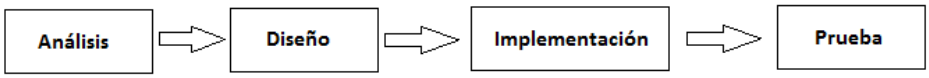


Ilustración 17: Proceso de desarrollo por etapas [27]

El desarrollo incremental del proyecto debe desarrollarse en etapas claramente diferenciadas que se exponen a continuación:

- **Análisis**: En esta fase de deben determinar los objetivos, las metas, requisitos y restricciones que pueda tener el proyecto a desarrollar

- **Diseño**: Describe cómo el sistema satisface los requisitos de la etapa anterior. Se describe los componentes que formarán parte del software con diferentes grados de detalle (alto o bajo).

- **Implementación**: Es la etapa donde se codifica el software. La programación puede distribuirse entre varios programadores dependiendo del tamaño del proyecto.

- **Prueba**: Mediante los resultados obtenidos debe valorase la calidad del mismo.

Todas las anteriores etapas vienen implícitamente ligadas a estas actividades, donde se exponen a continuación aquellas que no han sido detalladas anteriormente:

- **Ingeniería del sistema**: Es un modo de enfoque interdisciplinario para comprender y estudiar la realidad. Su propósito es implementar sistemas complejos.

- **Utilización**: Una vez la fase de prueba se ha realizado, el software debe pasar la fase de validación y verificación para que el sistema cumpla en su totalidad con su función

específica. El software debe estar instalado en un ambiente de explotación para realizar esta fase.

- **Mantenimiento**: Se verifica que el software funciona correctamente en su entorno de uso y en el caso de existir defectos o errores, proceder a corregirlos.

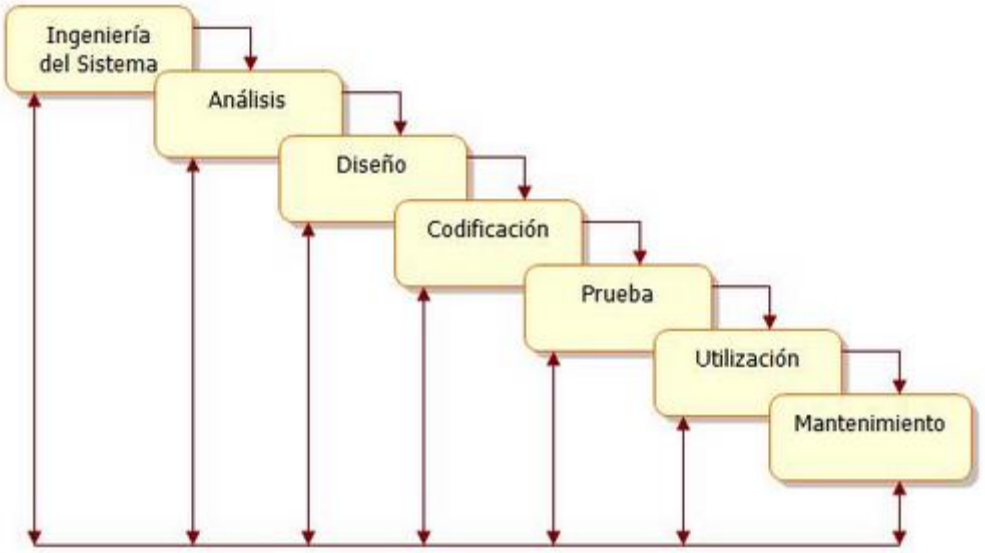


Ilustración 18: Actividades por etapas

4.2…1 Análisis

Una vez se ha definido el problema a tratar y el proceso de desarrollo del mismo, debe comenzar la fase de análisis.

Ha de tenerse en cuenta que la fase de análisis es de las fases más decisivas en todo el proyecto, por lo que deberá realizarse con mucha intensidad. Se dedica mucha atención a conocer la profundidad de los datos que se deben manejar, cuál será la función que debe cumplir el software, las interfaces requeridas o el rendimiento que se espera lograr.

Se deberá trabajar con el problema planteado por el cliente, se realizará una tabla de requisitos según el mismo y se analizarán todas las posibles alternativas que mejor se puedan ajustar a la solución.

El objetivo de esta sección es especificar el sistema, de forma detallada, que se plantea construir para resolver el problema. Es por ello que el análisis establece todas las necesidades y condiciones que debe cumplir el sistema que se quiere desarrollar.

En los siguientes apartados se deberá exponer todos los requisitos del sistema, que en este caso han sido desarrollados sin la necesidad del cliente ya que vienen impuestos por el tutor del TFG, por lo que no es necesario reunirse con el cliente para tratar y extraer toda la información sobre el funcionamiento del sistema.

Una vez se han definido los requisitos, será necesario analizar y especificarlos desde el punto de vista de la estructura, comportamiento y funcionalidad del sistema. A raíz de esto, se recogerán todas las necesidades definidas por los requisitos en forma de casos de uso

- *Definición de requisitos*

Los requisitos, tanto del sistema como del software deben ser documentados y revisados por el cliente, en este caso por el tutor del TFG. Mediante la fase de análisis se obtiene la especificación de estos requisitos software y puede ser mejorado con diagramas y descripciones en lenguaje natural.

Todo el conjunto de requisitos debe ser completo y conciso, tanto como sea posible para detallar claramente todas las funcionalidades y posibles restricciones que presenta el sistema que se quiere desarrollar. Es por ello que existen diferentes versiones de requisitos una vez se avanza en el desarrollo del software ya que está sujeto a modificaciones, pero aquí se ha querido plasmar la versión final de los mismos.

Los requisitos se han dividido en dos tipos: Funcionales y no funcionales.

- **Requisitos Funcionales**: Son los que definen el comportamiento interno del sistema. Definen una función del sistema software o de sus componentes, descrita como un conjunto de entradas, comportamientos y salidas. Dentro de los requisitos funcionales encontramos los cálculos, detalles técnicos, manipulación de datos y otras posibles funcionalidades específicas que cualquier sistema debe cumplir.

- **Requisitos no Funcionales**: Es el tipo de requisito que especifica aquellos criterios que se usan para juzgar la operación de un sistema, en lugar del comportamiento específico como hacen los requisitos funcionales. Especifican todos aquellos requisitos que no describen información a guardar o funciones a realizar. Hay que decir que este tipo de requisito está más orientado al diseño e implementación del sistema, por lo que se utilizan para limitar aquellos requisitos funcionales imponiendo una serie de condiciones sobre los mismos.
  Como ejemplo tenemos a los requisitos de rendimiento, accesibilidad, portabilidad, escalabilidad o usabilidad del sistema.

Para especificar cada uno de los requisitos se ha utilizado la plantilla de Volere [28] modificada para el tratamiento de los requisitos según nuestras exigencias en el proyecto. Campos como tipo de requisito o justificación de requisito se han eliminado ya que no se ha creído conveniente utilizarlos si se disponen de otros campos que cumplen la misma función. En el caso de tipo de requisito disponemos de identificador, y justificación de requisito sería siempre el tutor pero ya sabemos que los requisitos vienen impuestos por el propio objetivo del proyecto.

La satisfacción del cliente es otro de los puntos que se ha eliminado al no otorgar gran información al respecto, y otros campos como historia, materiales y conflictos se han eliminado por la misma razón.

Por lo tanto, la plantilla de *Volere* modificada quedaría de la siguiente forma:

Identificador	***R<Tipo de requisito>-<Numero>***		
Prioridad		**Necesidad**	
Descripción			
Prueba asociada			
Caso de uso asociado			

Tabla 25: Plantilla Volere modificada para los requisitos

Dónde:

- **Identificador**: Valor que se asigna al requisito para diferenciarlo. Está compuesto por:
  - Tipo de requisito: Diferenciará entre funcional o no funcional. En el caso de ser un requisito funcional se asignará una "F", mientras que si es el caso contrario se asignará como "NF".
  - Numero: Identifica numéricamente al requisito en cuestión

- **Prioridad**: Establece un grado de importancia sobre el cumplimiento del requisito frente a los restantes. Su rango de valores está definido como Alta, Media o Baja.

- **Necesidad**: Es el grado que se le exige al requisito, siendo el rango de valores Esencial, Opcional o Deseable.

- **Descripción**: Breve explicación del requisito

- **Prueba asociada**: Deben realizarse una serie de comprobaciones para cumplir el requisito

- **Caso de uso asociado**: Al finalizar los requisitos se incluyen unos casos de uso, el cual debe ser asociado al requisito según convenga

- *Especificación de requisitos*

Una vez se dispone de la plantilla asociada a los requisitos, se debe comenzar a detallar cada tipo de requisito según las exigencias del proyecto. Dado que la tabla de requisitos es muy extensa, se expondrá una breve definición del requisito en esta sección para dejar la especificación completa en los anexos

➢ Requisitos Funcionales

IDENTIFICADOR	DESCRIPCION
RF-1	La ventana principal del interfaz debe comprender dos secciones claramente diferenciadas: Listado Hadoop y visualización TextBox con los campos de los ficheros
RF-2	El usuario no dispondrá de ningún fichero en HDFS hasta que no se realice la primera carga de ficheros
RF-3	El botón “File” abre un nuevo navegador (File Search) para interactuar con el árbol de ficheros del propio disco duro interno
RF-4	File Search dispone de un árbol de ficheros del disco duro y una previsualización del fichero seleccionado
RF-5	Los botones de acceso en File Search solo estarán disponibles si se selecciona un fichero del árbol
RF-6	Cuando se selecciona un fichero en File Search y se clica sobre el botón “LOAD”, el fichero se copiará a la ruta preestablecida de Hadoop como un fichero HDFS
RF-7	El fichero copiado en File Search debe estar disponible en el listado Hadoop del interfaz principal de la aplicación
RF-8	El botón “View” de File Search permite visualizar parte del contenido del fichero seleccionado en el explorador de ficheros de Windows. La previsualización estará disponible en el cuadro de texto de la parte derecha del navegador File Search
RF-9	Cuando se previsualiza un fichero en Fil Search se habilitará el botón de “Show me more…”
RF-10	El botón “Show me more” permite abrir el bloc de notas con el contenido total del fichero seleccionado
RF-11	El botón “Reload” de File Search permite refrescar el explorador de ficheros de Windows y volver a la posición inicial desde el principio. Debe estar disponible en cualquier momento
RF-12	Pulsando nuevamente sobre el botón “File” de la barra de herramientas se cerrará el navegador de File Search
RF-13	Es posible cerrar el navegador File Search mediante el aspa ubicada en la parte superior derecha del mismo
RF-14	Cuando seleccionamos un fichero del listado principal de Hadoop en el interfaz principal y clicamos sobre el botón “Delete File”, el fichero se elimina del listado y de la ubicación Hadoop establecida.
RF-15	Cuando se selecciona el botón “MapReduce” de la barra de herramientas, se

	abrirá un nuevo navegador denominado MapReduce
RF-16	En el navegador MapReduce se establecen tres listados: Files, Key/s y Value/s
RF-17	El listado Files del navegador MapReduce contiene todos los ficheros cargados en la ubicación Hadoop y son seleccionables.
RF-18	Los listados Key/s y Value/s estarán deshabilitados hasta que no se seleccione un fichero del listado Hadoop
RF-19	Cuando seleccionamos un fichero Hadoop del navegador MapReduce se habilitan los listados Key/s y Value/s
RF-20	Los listados Key/s y Value/s del navegador MapReduce contienen todos los campos estructurados del fichero seleccionado
RF-21	Es posible seleccionar más de un valor en los listados Key/s y Value/s del navegador MapReduce
RF-22	Una vez se selecciona el fichero, la key/s y el value/s de un fichero Hadoop en el navegador MapReduce, es posible elegir entre las opciones “All Files” y “Selected File”
RF-23	Cuando se selecciona cualquier radioButton del navegador MapReduce se habilita el botón “MapReduce”
RF-24	Si se clica sobre el botón “MapReduce” del navegador MapReduce saltará un mensaje para visualizar o no el fichero generado por el paradigma MapReduce
RF-25	Cuando se selecciona visualizar el fichero generado por el paradigma MapReduce, se abrirá el bloc de notas para visualizar el resultado obtenido por el paradigma.

Tabla 26: Requisitos funcionales

- Requisitos No Funcionales

IDENTIFICADOR	DESCRIPCION
RNF-1	El entorno sólo podrá ser operativo si se cuenta con una instalación del framework Hadoop
RNF-2	Es necesario disponer de las librerías Hadoop necesarias para lanzar la aplicación
RNF-3	El estándar aplicado al entorno visual es el X-Windows de uso generalizado
RNF-4	La aplicación estará optimizada para cualquier versión de Windows actual (a partir de Windows XP)
RNF-5	La aplicación solo estará disponible con el idioma Ingles

RNF-6	La aplicación está pensada para una resolución mínima de 1024x768 pixeles
RNF-7	La proporción de los tamaños de los elementos debe ser estática y no se dimensionará según la resolución de pantalla
RNF-8	Solo estará disponible un resultado del paradigma MapReduce. Si el usuario lanza nuevamente el paradigma, se borrará el anterior resultado obtenido
RNF-9	No se guardará ningún historial referente a la anterior ejecución del programa

Tabla 27: Requisitos no funcionales

- *Casos de uso*

Los casos de uso intentan plasmar los pasos necesarios que debe seguir el usuario para llevar a cabo algún tipo de proceso [29]. En nuestro caso, sirven para definir todos los pasos que el usuario debe seguir para cumplir con la funcionalidad descrita en los requisitos plasmados con anterioridad.

Los pasos se definen a partir de la interacción realizada entre el usuario y el sistema.

Se ha diseñado una plantilla para representar cada caso de uso, y sigue la siguiente estructura:

Caso de uso	**Identificador**	**CU-*<Tipo de aplicación>*-*<Numero>***
Actores		
Descripción		
Precondiciones		
Pasos a seguir		

Tabla 28: Plantilla casos de uso

El significado de cada campo de la plantilla se detalla a continuación:

- Caso de uso: Nombre que identifica a cada caso de uso
- Identificador: Para poder identificar a cada caso de uso de manera única. Se basa en la estructura CU más el numero identificativo del caso de uso. Por ejemplo, CU-12
- Actores: Detalla la entidad externa al propio sistema y la cual demanda la funcionalidad descrita por el caso de uso
- Escenario: Especifica el contexto de uso del caso
- Precondiciones: Son todas aquellas condiciones que debe cumplirse para poder ejecutar el caso de uso
- Pasos a seguir: Se identifican cada uno de los pasos necesarios para llevar a cabo el caso de uso

A continuación se exponen todos los casos de uso:

Caso de uso	**Cargar Fichero a Hadoop**	**Identificador**	**CU - 1**
Actores	Usuario		

Descripción	El usuario abre la ventana File para cargar un fichero de su disco duro al sistema de ficheros HDFS y así poder trabajar con él.
Precondiciones	Ninguna
Pasos a seguir	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior de la aplicación 3. Seleccionar un fichero del disco duro del árbol de ficheros de la ventana File Search 4. Clicar sobre el botón LOAD para disponer del fichero en HDFS

Tabla 29: Caso de uso 1

Caso de uso	**Eliminar un fichero del listado Hadoop**	**Identificador**	**CU - 2**
Actores	Usuario		
Descripción	El usuario debe borrar un fichero del listado HDFS		
Precondiciones	Haber cargado previamente al menos un fichero en HDFS		
Pasos a seguir	1. Arrancar la aplicación 2. Seleccionar un fichero del listado Hadoop de la ventana principal de la aplicación 3. Se habilita el botón Delete de la parte inferior del listado Hadoop 4. Clicar sobre dicho botón y el fichero desaparece del listado		

Tabla 30: Caso de uso 2

Caso de uso	**Visualizar campos de un fichero Hadoop**	**Identificador**	**CU - 3**
Actores	Usuario		
Descripción	El usuario desea comprobar los campos por los cuales se estructura el fichero seleccionado		
Precondiciones	Haber cargado previamente al menos un fichero en HDFS		
Pasos a seguir	1. Arrancar la aplicación 2. Seleccionar un fichero del listado Hadoop de la ventana principal de la aplicación 3. Doble click sobre el fichero seleccionado 4. Aparece un textBox con la estructura del fichero a modo de listado en la parte derecha de la ventana principal de la aplicación		

Tabla 31: Caso de uso 3

Caso de uso	Realizar un paradigma MapReduce simple	Identificador	CU - 4
Actores	Usuario		
Descripción	El usuario desea realizar MapReduce sobre un único fichero y con un único campo Key y otro tipo Value		
Precondiciones	Haber cargado previamente al menos un fichero en HDFS		
Pasos a seguir	1. Arrancar la aplicación 2. Clicar sobre el botón MapReduce de la barra de herramientas superior 3. Seleccionar un fichero del listado HDFS en la nueva ventana abierta 4. Seleccionar un único valor Key del listado Key/s 5. Seleccionar un único valor Value del listado Value/s 6. Seleccionar el radioButton File selected 7. Lanzar MapReduce con el botón ahora habilitado		

Tabla 32: Caso de uso 4

Caso de uso	Realizar un paradigma MapReduce múltiple	Identificador	CU - 5
Actores	Usuario		
Descripción	El usuario desea realizar MapReduce sobre todos los ficheros HDFS, con varios valores Key y Value		
Precondiciones	Haber cargado previamente al menos dos ficheros en HDFS con la misma estructura interna		
Pasos a seguir	1. Arrancar la aplicación 2. Clicar sobre el botón MapReduce de la barra de herramientas superior 3. Seleccionar un fichero del listado HDFS en la nueva ventana abierta 4. Seleccionar varios valores Key del listado Key/s pulsando ctrl+click 5. Seleccionar varios valores Value del listado Value/s pulsando ctrl+click 6. Seleccionar el radioButton All Files 7. Lanzar MapReduce con el botón ahora habilitado		

Tabla 33: Caso de uso 5

Una vez expuestos los requisitos y los casos de uso de la aplicación será necesario realizar la matriz de trazabilidad entre ellos. La finalidad de la tabla es correlacionar estos dos aspectos para determinar la integridad en la relación, es decir, tenemos que mostrar que todos los requisitos están asociados a un caso de uso y que a su vez cada caso de uso puede disponer de más de un requisito. Si se cumple lo anterior, la matriz de trazabilidad será correcta.

A continuación se adjunta la matriz de trazabilidad entre los requisitos y los casos de uso expuestos anteriormente:

REQUISITO	CU-1	CU-2	CU-3	CU-4	CU-5
RF-1	X	X	X	X	X
RF-2	X	X	X	X	X
RF-3	X	X	X		
RF-4	X	X	X		
RF-5	X		X		
RF-6	X				
RF-7	X	X	X		
RF-8			X		
RF-9			X		
RF-10			X		
RF-11			X		
RF-12	X	X			
RF-13	X	X			
RF-14		X			
RF-15				X	X
RF-16				X	X
RF-17				X	X
RF-18				X	X
RF-19				X	X
RF-20				X	X
RF-21				X	X
RF-22				X	X
RF-23				X	X
RF-24				X	X
RF-25				X	X
RNF-1	X	X	X	X	X
RNF-2	X	X	X	X	X
RNF-3	X	X	X	X	X
RNF-4	X	X	X	X	X
RNF-5	X	X	X	X	X
RNF-6	X	X	X	X	X
RNF-7	X	X	X	X	X

RNF-8	X	X	X	X	X
RNF-9	X	X	X	X	X

Tabla 34: Matriz de trazabilidad requisitos-casos de uso

Se aprecia que todos los requisitos están presentes en al menos un caso de uso en concreto, lo que significa que todos los posibles casos de uso están trazados correctamente.

4.2…2 Diseño

La fase de diseño se basa en cuatro características distintas

- **Estructura de los datos**
- **Arquitectura de las aplicaciones**
- **Estructura interna del programa**
- **Interfaces**

Esta fase debe traducir todos los requisitos previos de forma que se conozca la arquitectura, funcionalidad y la calidad del producto antes de comenzar con la implementación de código.

Es por ello que la fase de especificación de requisitos cobra especial importancia en el diseño, ya que se traducen a diagramas que pueden representar la estructura del sistema software, de los datos e interfaces desarrolladas.

La fase de diseño tiene como principal objetivo guiar el proceso de producción del sistema, generando modelos de entidades que se construirán posteriormente. La fase de análisis es muy importante en el diseño ya que todos los requisitos establecidos anteriormente deben cumplirse para la implementación del sistema, ajustándose a las condiciones dadas por el propio sistema.

En esta fase se detallará todo el seguimiento del modelo de proceso de desarrollo, el modelo basado en prototipos de distinto nivel hasta llegar a un diseño fiable y que cumpla con la calidad expuesta por el cliente. Es posterior a la fase de análisis y comienza especificando el contexto en el que se basa el sistema a diseñar para más adelante exponer la arquitectura adoptada y los componentes software necesarios para cumplir con la funcionalidad del sistema.

- *Contexto del sistema*

Antes de comenzar con el diseño de la aplicación es importante tener una visión general del sistema para así poder conocer el marco en el cual se encuentra, los sistemas que interactúan en el mismo y sus límites.

Para este caso, el usuario es el que proporciona la entrada al sistema seleccionando los ficheros para posteriormente manejarlos en la aplicación. Como se ha comentado con

anterioridad, no todos los ficheros del disco duro están disponibles para manejarse en la aplicación ya que estos deben cumplir un estándar .csv.

La estructura de estos ficheros tiene que ser seguida por el estándar aplicado, sino es posible que los resultados obtenidos en la fase de MapReduce o visualización del contenido de los campos no sea correcta.

La finalidad de la aplicación es disponer de un interfaz sencillo y visualmente cómodo para trabajar con el sistema de ficheros HDFS, por lo que cuanto mejor sea la interacción entre ficheros Windows y Hadoop más cómodo se sentirá el usuario.

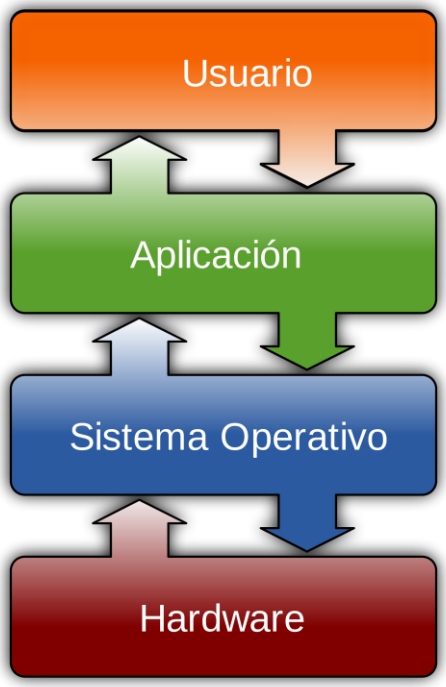
Los recursos generados por el sistema son los siguientes:

- Visualización total de los campos del fichero seleccionado
- Resultados del paradigma MapReduce sobre la consulta realizada

Estos dos recursos pueden ser utilizados de múltiples maneras por el usuario. Es posible conocer los campos del fichero seleccionado para comprobar si cumple con lo deseado, si es el fichero correcto para relacionar con otro fichero o simplemente para conocer la estructura del mismo.

De igual manera los resultados proporcionados por MapReduce pueden servir para comprobar correlaciones entre diferentes campos, relaciones entre diferentes ficheros que no se había establecido hasta ahora o proporcionar mejoras relativas a los datos utilizados. Eso ya se lo podemos dejar al cliente, ya que no está diseñada la aplicación para generar una única solución sino que es el cliente el que establece los límites del paradigma para su uso propio.

La siguiente ilustración muestra la interacción entre los diferentes elementos que entran en juego cuando se ejecuta la aplicación:



Cuando el usuario arranca la aplicación e interactúa con ella, se pone en proceso tanto el sistema operativo como el hardware del ordenador.

Un usuario tiene la capacidad de dar la entrada a la aplicación con los ficheros que seleccione, mientras que la aplicación para devolver resultados debe interactuar con el sistema operativo y el hardware del sistema, que son los encargados de generar el resultado pedido por el usuario.

Si el proceso es completo por parte de la aplicación y sus niveles inferiores, ésta proporcionará un fichero con los resultados del paradigma o un *textbox* con los campos de fichero, eso ya dependerá de la entrada proporcionada por el usuario.

Ilustración 19: Elementos en la ejecución de la aplicación

- *Diseño de sistema*

Este apartado de diseño presenta la arquitectura del sistema y todas las tecnologías utilizadas en el desarrollo del proyecto.

El objetivo de seleccionar una arquitectura del sistema es tener una visión general de la estructura que llevará nuestro sistema, conociendo todos los componentes y su interacción entre ellos.

Se debe tener en cuenta las herramientas para el desarrollo del sistema, que en este caso serán con base Java y un framework específico para el interfaz de usuario denominado *Java Swing*.

Este framework se caracteriza por ser una biblioteca gráfica para un entorno Java, la cual incluye widgets para la interfaz de usuario como pueden ser *Jframe*, *JLabel*, *JButtons* o *JTables*.

Lo bueno de *Java Swing* es que permite una independencia total con la plataforma [30], siguiendo una programación por hilos cuyas características se detallan a continuación:

- Independencia de plataforma
- Arquitectura altamente particionada para que los usuarios puedan crear sus propias implementaciones modificadas de las originales. Es posible además extender las clases existentes para proveer alternativas en el desarrollo de la aplicación.
- Altamente personalizable: Permite representar diferentes estilos de apariencia "*look and feel*", permitiendo cambios en la apariencia de la aplicación sin efectuar cambios en el código.

Dentro de las ventajas que proporciona *Java Swing* cabe destacar las siguientes [31]:

- Un diseño puro en Java proporciona limitaciones en la plataforma que *Java Swing* sí que puede proporcionar
- Los componentes *Swing* pueden desarrollarse de forma más activa
- El diseño de componentes en *Swing* soporta más características que Java

Dado que *Java Swing* es un framework, será necesario incluirlo en una plataforma de desarrollo que en este caso será *Eclipse* en su versión *Kepler*. Se ha utilizado esta versión con el fin de interactuar con *Swing* de la manera más sencilla y a su vez que fuese compatible con el entorno Hadoop.

No se cree necesario exponer *Eclipse* en su totalidad, ya que existen multitud de plataformas para desarrollo Java y se ha utilizado ésta al ser la que tiene interacción con Hadoop. Además ésta ha sido la herramienta utilizada a lo largo de toda la carrera para desarrollo Java.

Para hacerse una idea de la estructura de este framework se ha querido plasmar una captura del mismo incluyendo las distintas secciones que proporciona.

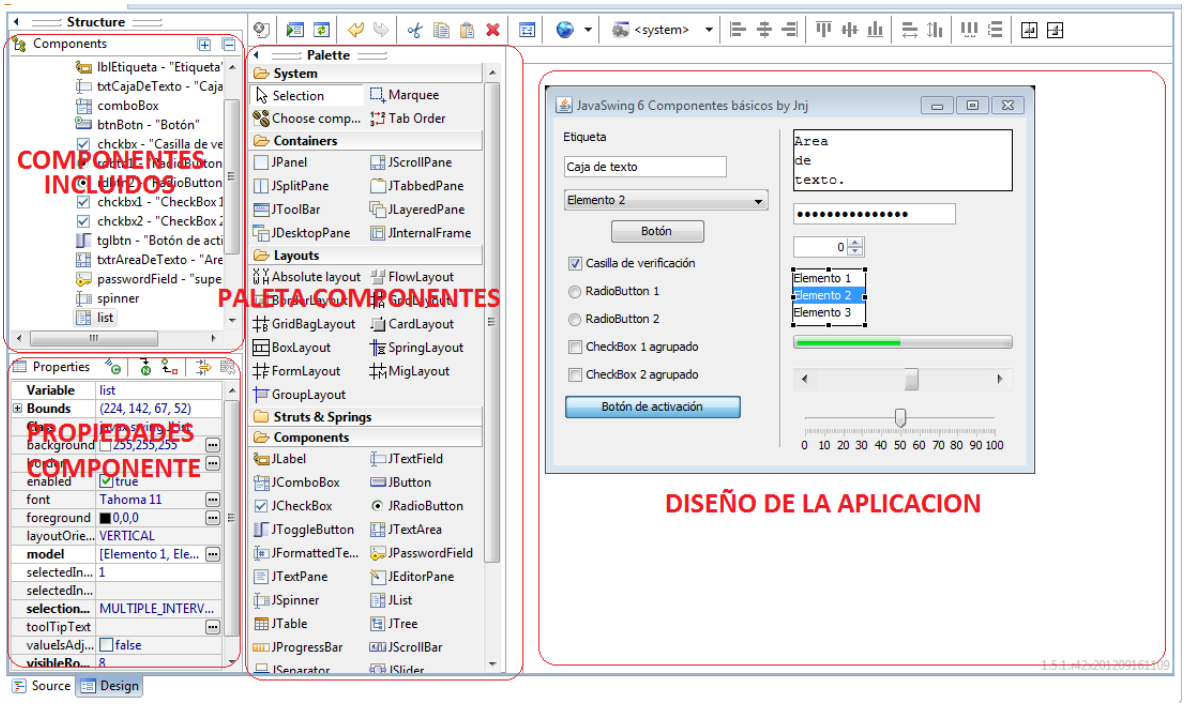


Ilustración 20: Captura de pantalla Java Swing

El framework *Java Swing* utiliza el patrón de diseño *MVC*, por lo que se cree conveniente utilizar dicho patrón para la arquitectura de la aplicación [32].

No está de más recordar este patrón:

- **Modelo**:
  Se denomina modelo al conjunto de cases que representan la información del mundo real que el sistema debe procesar, sin tener en cuenta ni la forma en la que la información será mostrada ni los mecanismos necesarios que hacen que esos datos estén dentro del modelo. Es decir, no dispone de ninguna relación entre entidades de la propia aplicación.
  El modelo no reconoce la existencia de las vistas ni del controlador, pero realmente en la práctica deben existir interfaces que permitan a los módulos comunicarse entre sí.

- **Vista**:
  Las vistas son todo el conjunto de clases cuya finalidad es mostrar al usuario final la información contenida en el modelo. Por lo tanto, una vista está asociada a un modelo en concreto, pudiendo existir más de una vista para cada modelo. Por ejemplo, es posible tener una vista de la hora en reloj analógico y en un reloj digital al mismo tiempo.

Una vista asociada a un modelo obtiene únicamente la información necesaria para desplegar y actualizarse en base a ese modelo, por lo que si éste cambia, la vista también lo hará.

- **Controlador**:
  Es el encargado de dirigir el flujo de control de la aplicación en base a mensajes externos, como son los datos introducidos por el usuario u otras opciones del menú que contenga nuestra aplicación. A partir de ello, el controlador debe ser capaz de modificar el modelo e interactuar con las vistas teniendo siempre acceso a los mismos, cosa que tanto la vista como el modelo desconocen de la existencia del controlador.

Esta división por capas permite separar la lógica de la aplicación de la interfaz, lo que proporciona modularidad a la vez que se minimizan las dependencias que puedan existir entre las diferentes partes. Es por ello que el *MVC* facilita el mantenimiento, aumentando la capacidad de reutilización de las partes del sistema para así poder otorgar un plus de flexibilidad a la aplicación desarrollada.

Para tener una idea clara del concepto *MVC* se adjunta la siguiente ilustración:

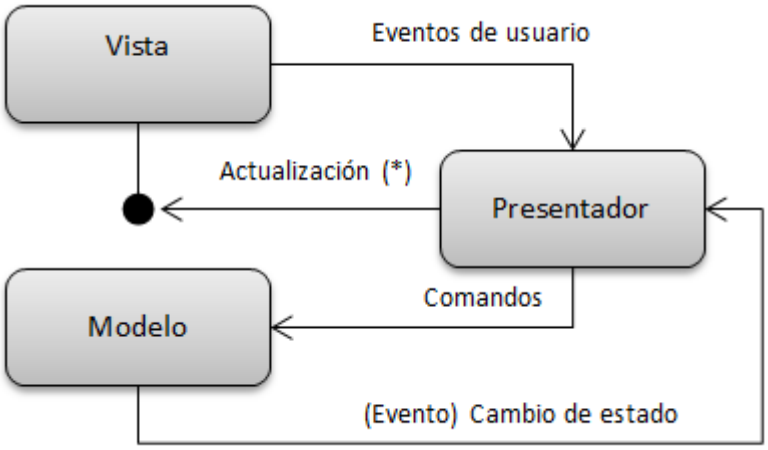


Ilustración 21: Esquema Modelo Vista Controlador [33]

Este tipo de arquitectura mostrada es una de las variantes existentes en MVC y se denomina *MVP como vista pasiva*. Su variante se basa en el concepto de vista "tonta" que en este caso no tiene ningún tipo de lógica y cuya única funcionalidad es mostrar toda la información que se le pasa a través del interfaz Vista. Cuando un usuario realiza una operación sobre la interfaz, la vista delega directamente todos los eventos al Presentador (Controlador). Éste será el encargado de realizar los cambios sobre la vista si fueran necesarios y realizará las llamadas a los comandos para llevar a cabo la operación solicitada por el usuario.

Otra de las variantes de este *MVC* es el Modelo. Cuando éste genere cambios en su estado, el Presentador será el encargado de recogerlos para solicitar nuevamente al Modelo los cambios solicitados y así poder actualizar la Vista acorde a esos mismos cambios recibidos.

Dado que el componente Vista en la aplicación desarrollada es necesario exclusivamente para mostrar los cambios solicitados por el usuario, se ha decidido utilizar este tipo de arquitectura modificada.

Para generar el diagrama de componentes de la aplicación es necesario disponer de los requisitos del apartado de análisis, agrupándolos según su funcionalidad y asignando estos grupos a los componentes que se encarguen de esa misma funcionalidad.

Una vez se dispone de los componentes, el siguiente paso es representar las relaciones existentes entre ellos que serán las comunicaciones entre los mismos. La división final de componentes se muestra en la siguiente ilustración:

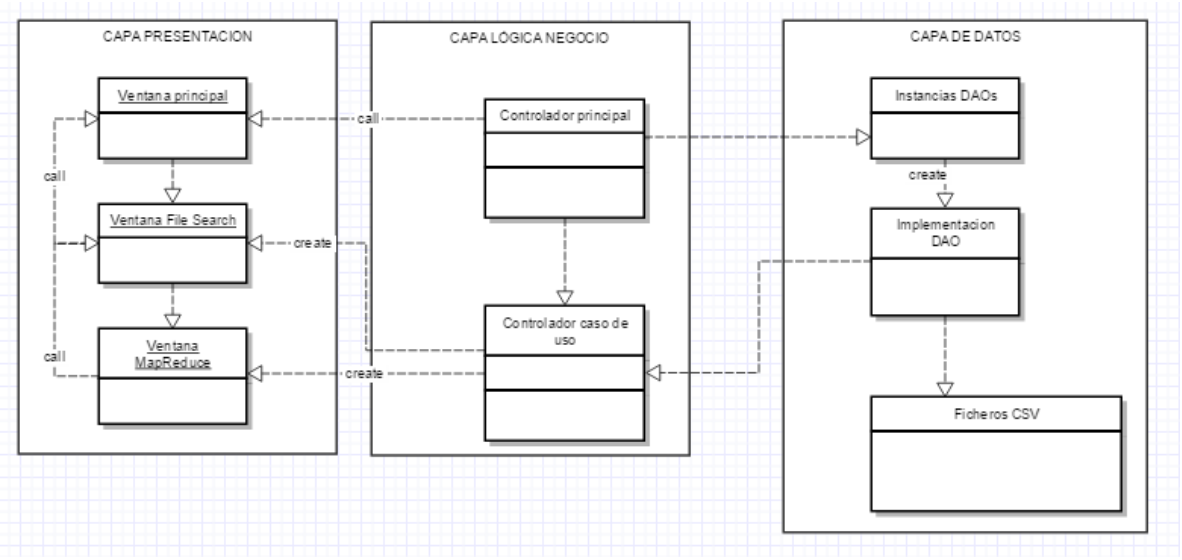


Ilustración 22: Diagrama de componentes

- *Diseño detallado*

En este apartado se mostrará el diseño final de la aplicación, teniendo en cuenta que los cambios en el diseño a lo largo de todo el desarrollo han sido mínimos y de poca importancia como pueden ser tamaño del interfaz o la reubicación de botones.

Dichos diseños han sido realizados con la herramienta Pencil, un software libre para el diseño de interfaces de manera sencilla. Esta aplicación es muy rica en componentes visuales al estilo Windows, por lo que se ha decidido utilizarla dado que el estilo de Java Swing refleja el del sistema operativo Windows. La técnica que utiliza es la de drag-and-drop, muy utilizada en este tipo de programas para arrastrar los componentes al diseño de nuestra aplicación.

El diseño de la aplicación de la herramienta ha sido diseñado desde cero, sin ningún tipo de plantilla auxiliar.

Para detallar las diferentes ventanas de la aplicación de manera exhaustiva se irá en proceso de los botones auxiliares de nuestra barra superior de herramientas, de izquierda a derecha.

Todas las capturas que se mostrarán a continuación tienen su propia explicación para entender el funcionamiento total de la aplicación, incidiendo en los apartados más costosos del desarrollo del mismo.

Comenzaremos con la interfaz principal para acabar con la salida del programa, todo ello mostrado con el prototipo final de la aplicación.

➢ Pantalla principal

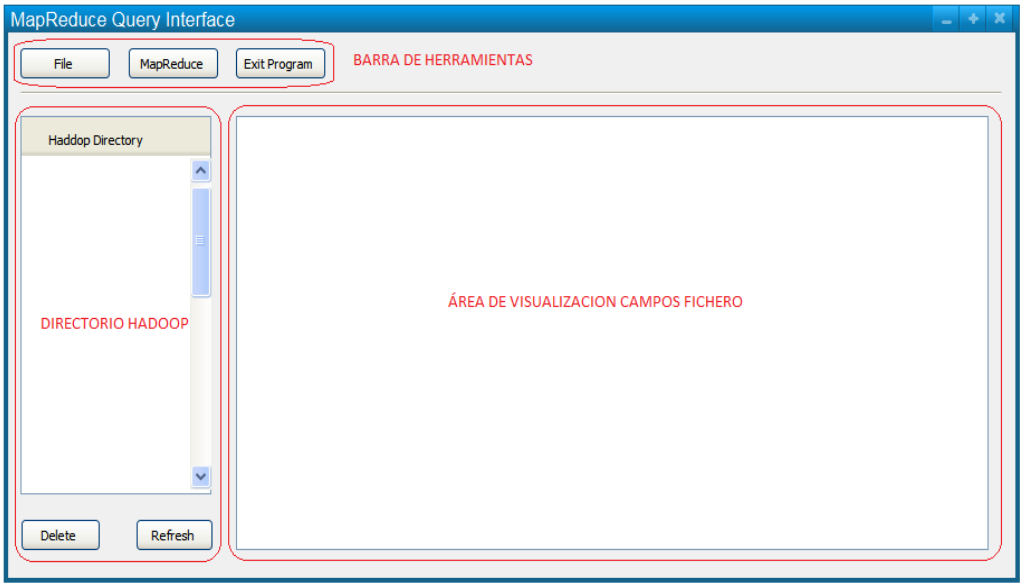


Ilustración 23: Pantalla principal

La pantalla principal del programa está dividida en tres secciones claramente diferenciadas.

Primero disponemos de una barra de herramientas superior con tres botones, los cuales distribuyen el funcionamiento de la aplicación. El primero de ellos es "File" que permite seleccionar un fichero de nuestro directorio Windows para poder manejarlo en un futuro por Hadoop. El siguiente botón es el de "MapReduce", núcleo principal de la aplicación y el que maneja todo lo relativo al paradigma que lleva su nombre.

Por ultimo tenemos el botón de "Exit program" para finalizar la sesión en la aplicación y salir de la misma.

Todas las funcionalidades propias de estos botones se detallarán en sus respectivas capturas del prototipo, detallando todo lo relativo a su funcionamiento, cometidos y distribución de su propia ventana de navegación.

La siguiente sección es la de Directorio Hadoop. Esta sección permite visualizar todos los ficheros que tengamos cargados en la ubicación Hadoop a modo de listado para posteriormente manejarlos en la herramienta. El manejo completo de ficheros, como puede ser su carga o visualización se encarga la ventana de navegación de File, que se hablará más adelante en su captura de pantalla del prototipo.

El inconveniente de Hadoop para Java es que es extremadamente complicado programar un complemento que permita ubicarse y navegar por el directorio completo de Hadoop a modo de explorador de ficheros, por lo que en un principio se descarta esa idea para simplemente mostrar un listado de ficheros cuyo directorio Hadoop vendrá predefinido por la aplicación.

Cuando el usuario disponga de ficheros en el listado podrá seleccionarlos, borrarlos, actualizar el listado o mostrar la estructura del mismo en la sección de "Área de visualización de campos de fichero".

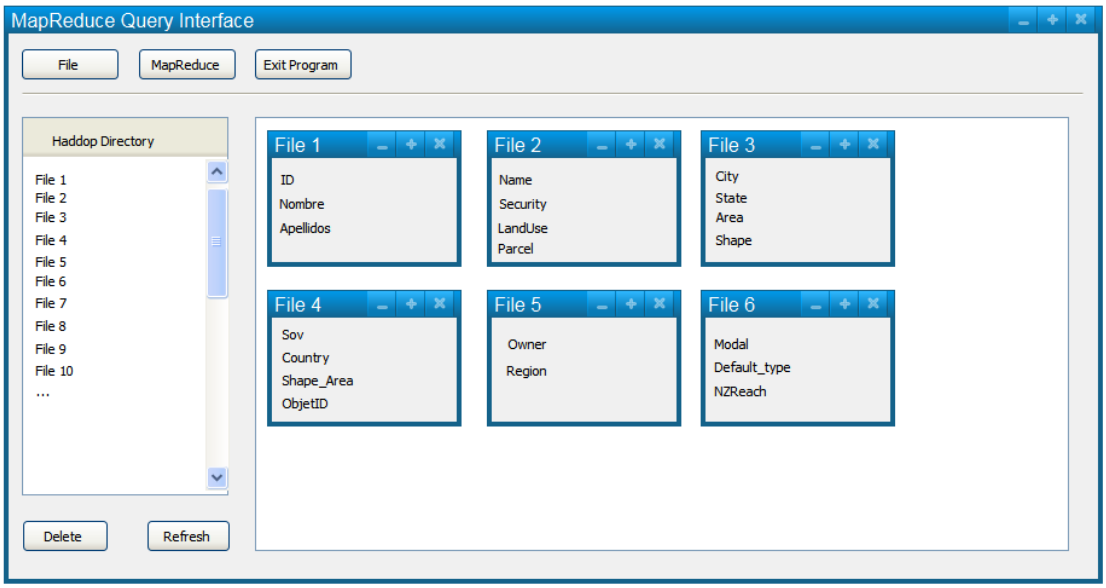


Ilustración 24: Pantalla principal con ficheros y sus campos

Llegamos a la última sección de la pantalla principal, cuyo objetivo es poder mostrar la estructura del fichero que se ha seleccionado del listado Hadoop. Cada visualización de fichero en esta sección se muestra con una pequeña caja de texto donde se especifica el fichero mostrado y la estructura del mismo. Esta estructura se basa en la cabecera principal del fichero .csv, y dado que se ha especificado como deben ser los ficheros de texto para esta aplicación, no resulta excesivamente completa la lectura de estos campos cuando se disponga la programación.

La idea principal de esta sección es tener clara la visualización de la estructura del fichero con el cual se quiere trabajar, como se puede relacionar con otros ficheros del listado o simplemente cerciorarse de que es ese el fichero con el cual se quiere realizar el MapReduce.

Para que el usuario disponga de una mayor libertad en esta sección, se ha querido otorgar redimensionalidad a las cajas de texto. Esto se traduce en una reubicación de cajas según quiera el usuario, para así poder distribuir en la sección los ficheros según como se quiera. El límite de ficheros a visualizar en esta área es limitado, no en número ya que se puede mostrar cajas hasta que uno quiera, sino de espacio. Se cree conveniente no cargar en exceso esta sección para poder tener una vista detalla de cada fichero que se quiera trabajar para no abrumarse con tantas cajas al mismo tiempo.

➢ File Search

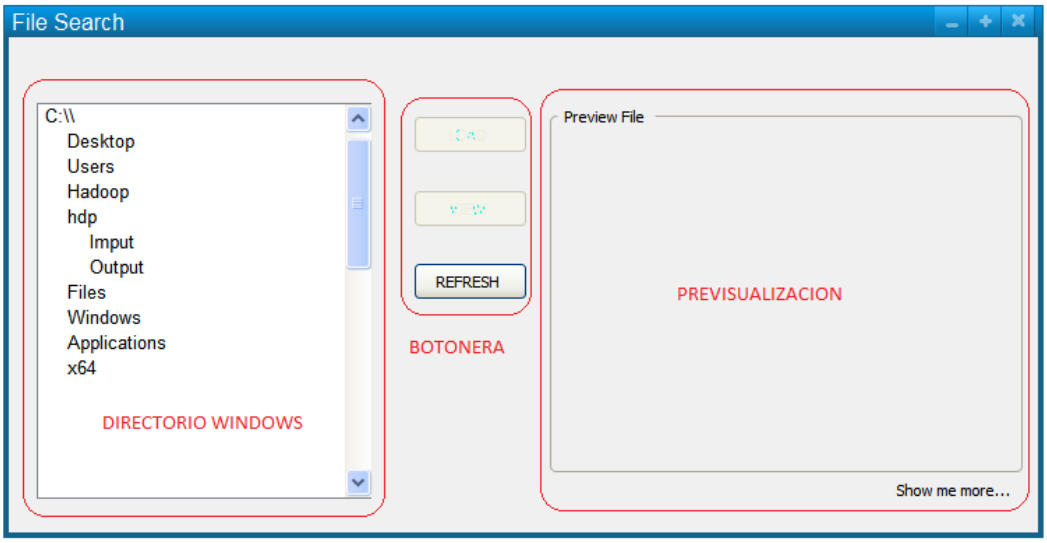


Ilustración 25: Ventana File Search

Llegamos a la primera ventana adicional de la aplicación. Ésta permite el manejo total de los ficheros con los cuales el usuario quiere trabajar en Hadoop. Anteriormente en la otra sección se

comentaba que existe un directorio principal para el almacenamiento de ficheros Hadoop, bien pues para cargar ese listado será necesario pasar previamente por esta ventana.

Para poder llegar hasta la misma, simplemente se debe clicar el botón de File ubicado en la barra de herramientas de la pantalla principal. Una vez se realice la acción, se abrirá automáticamente esta ventana de File Search.

Como se ha hecho con anterioridad, esta ventana también se detallará por medio de secciones diferenciadas.

La primera de ellas será la del Directorio Windows. Esta sección está desarrollada con un componente árbol, que permite la visualización completa del disco duro alojado en el ordenador, da igual si es externo o interno, se podrá navegar de igual forma por todo el contenido. Este componente es muy moldeable para su visualización, capaz de mostrar en diferentes formas el contenido, otorgando un plus de calidad con respecto a otro tipo de componentes que puedan llegar a realizar lo mismo. Lo bueno es que dando una ubicación base a mostrar, él solo lista todas las carpetas y su contenido en forma de ficheros. Además también permite ver la extensión del fichero, y dado que solo será posible trabajar con ficheros .csv, esto es muy cómodo a la hora de seleccionar un fichero conociendo que ese es el fichero correcto.

Como seguramente el listado total del disco duro sea extenso, se aplicará un componente barra deslizamiento para que el usuario pueda moverme libremente por el explorador y así poder visualizar todos los ficheros y carpetas sin excepción.

La siguiente sección es la botonera, compuesta por tres botones de los cuales dos de ellos estarán deshabilitados en un primer momento. Estos botones permiten el manejo del árbol de navegación de Windows hacia Hadoop.

El primero de ellos es el de carga de fichero. Una vez el usuario seleccione un fichero (y no un directorio) del árbol de navegación de la parte izquierda, se habilitará las opciones de “LOAD” y “VIEW”. El botón de LOAD permite cargar el fichero seleccionado al directorio Hadoop que se haya escogido previamente (por defecto se incluirá uno) para posteriormente poder trabajar con él en la ventana principal de la aplicación.

El segundo botón de esta sección es el de VIEW, que permite visualizar parte del contenido del fichero para comprobar que realmente es el fichero con el cual se quiere trabajar. El límite es una restricción necesaria para este apartado y es necesario tenerlo en cuenta cuando se realice la implementación del código ya que es posible encontrarse con ficheros de un peso relativamente grande (25mb) que si se tiene que mostrar en la sección de previsualización puede tardar un tiempo el cual no se está dispuesto a esperar. Por ello el límite de líneas mostradas será de 100, número más que suficiente para conocer el contenido del fichero y hacerse una idea de si es o no el fichero que se quiere escoger para trabajar con él.

Como se ha comentado con el botón de VIEW, será posible previsualizar el fichero seleccionado del árbol de Windows en la sección de previsualización como se muestra en la captura. Además, si el usuario no tiene suficiente con esta previsualización es posible mostrar la totalidad del contenido del fichero con el botón “Show me more” ubicado en la parte inferior derecha de esta sección.

Con ello se abrirá el bloc de notas de Windows y se mostrará el fichero al completo, pero siempre teniendo en cuenta que es un editor de texto básico, sin guardar formatos ni estilos por lo

que el diseño del fichero original puede ser muy distinto al que se muestre con este programa. No se cree un inconveniente, ya que esta aplicación está diseñada para trabajar con el contenido del fichero no con su diseño.

Una vez se realice alguna acción de carga de fichero, se podrá comenzar con el paradigma, el cual se detalla en el siguiente punto.

- Map Reduce

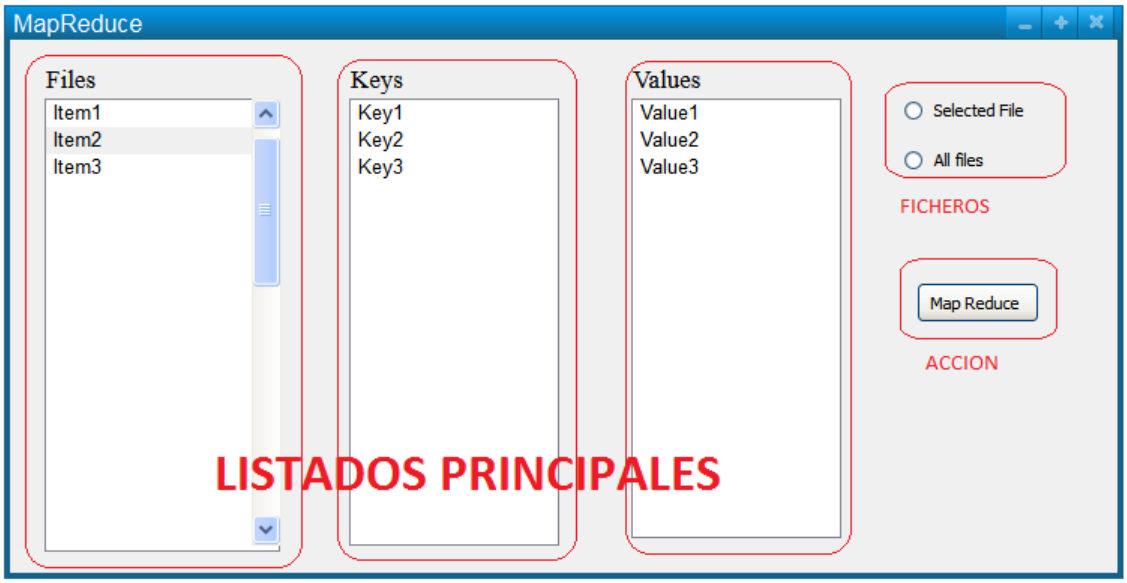


Ilustración 26: Pantalla MapReduce

Otra de las ventanas de navegación que encontraremos en la aplicación es la de MapReduce. La estructura será sencilla para no perderse en ella, por lo que se ha diseñado de manera minimalista y a la vez completa en cuanto a aspectos modificables del paradigma.

Es verdad que el propio paradigma deja poco a la imaginación y desarrollo de código, pero esta sección permitirá manejar cada aspecto relacionado con Map Reduce y el fichero que se quiere tratar.

Lo primero que hay que tener en cuenta es recordar cómo funciona MapReduce, sabemos que es necesario un fichero, una/s key y un value/s. Dado que eso es el cuerpo del paradigma, se ha querido retratar de igual manera en la ventana MapReduce. Si se observa la captura del prototipo se aprecia la sección con los listados principales, el manejo de ficheros mediante los radiobuttons y el botón de Map Reduce para lanzar el paradigma.

Los listados principales se dividen en Files, Keys y Values. Si el usuario ha realizado una carga previa de ficheros con el navegador File Search a la ubicación Hadoop, éstos se encontrarán todos en este listado. La idea es disponer de los ficheros Hadoop o ficheros .csv y seleccionar uno para lanzar el paradigma.

Toda esta sección está distribuida de izquierda a derecha y se irán abriendo nuevas acciones a medida que se completen las anteriores, es decir, si el usuario no elige un fichero no podrá ver el listado de Keys por ejemplo.

Una vez el usuario elige un fichero, automáticamente se cargarán todos los campos con los cuales está compuesto en el siguiente listado, Keys. Estos campos serán los mismos que los que se hayan podido cargar en la pantalla principal del programa, recordemos las cajas de texto dimensionables que se hablaba en el primer apartador de diseño detallado.

Al igual que con la acción de selección en el listado de ficheros, cuando se elija una o varias keys será posible elegir también uno o varios values para lanzar el paradigma.

Cuando el usuario haya decidido tanto el fichero, como sus posibles keys y values se abrirá la sección de fichero. Ésta sección quiere dar la oportunidad de lanzar el paradigma sobre todos los ficheros ubicados en el directorio Hadoop o simplemente sobre el que se haya elegido previamente.

Es obvio que si se selecciona todos los ficheros, cada uno de ellos en el directorio debe tener una estructura concreta y no variar en nada ya que si no es posible que el paradigma no trabaje bien y arroje resultados incoherentes. Es por ello que se quiere dejar claro que sólo funcionará un MapReduce múltiple si todos los ficheros del directorio tienen exactamente la misma estructura.

Se ha querido diseñar esta funcionalidad para aquellos ficheros que se tengan que generar cada X periodo de tiempo, como facturas, mediciones, contratos… que se sabe que siempre llevarán la misma estructura.

Por último solo queda lanzar el paradigma con el botón de “Map Reduce” ubicado en la parte inferior derecha de la ventana. Es importante comentar que esta acción solo estará disponible cuando se haya pasado por todas las selecciones previas, como selección de fichero, keys, values y numero de ficheros a tratar.

Cuando se lance el paradigma, todos los valores seleccionados serán los encargados de dar los resultados realizados por la implementación del código.

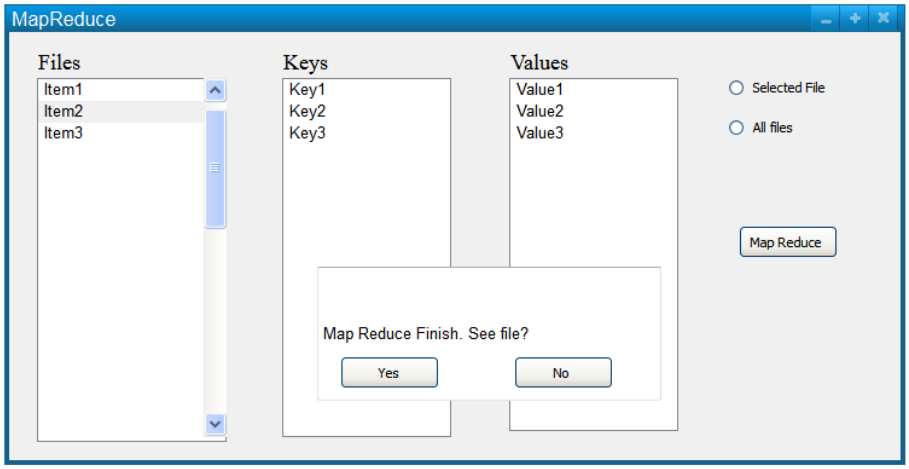


Ilustración 27: Pantalla resultado del paradigma MapReduce

Para conocer los resultados del paradigma una vez se realice la acción, se mostrará por pantalla un pop-up emergente permitiendo al usuario conocer el resultado o no en ese momento.

Si el usuario desea verlo, simplemente seleccionará el botón YES y se abrirá un bloc de notas con los resultados obtenidos. Según el tamaño del fichero generado se tardará más o menos en abrirlo.

- Exit Program

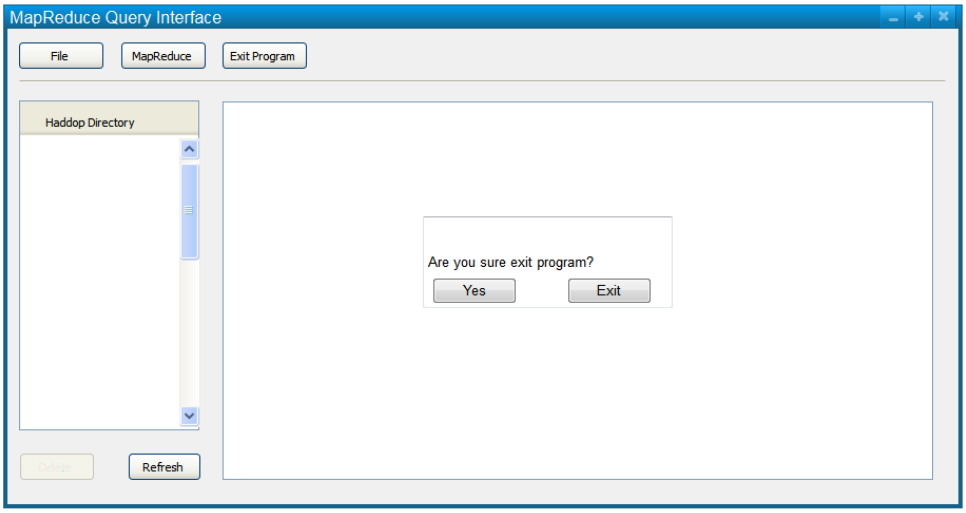


Ilustración 28: Pantalla Exit Program

La última sección a tratar en el diseño detallado es el cierre de la aplicación. Simplemente comentar que el último botón de la barra de herramientas superior permite al usuario cerrar la aplicación cuando ya no se quiera realizar ninguna acción más.

Para cerciorarse de que realmente es lo que se quiere hacer y no perder información útil de la sesión, se lanzará un pop-up preguntando si quiere o no cerrar la aplicación. En el caso de pulsar YES, el programa se cerrará y la sesión se perderá.

4.2…3 Implementación

La fase de implementación traduce el diseño a un formato máquina para su codificación. Si el diseño ha sido satisfactorio y altamente detallado, la implementación debe ser sencilla aunque no siempre será así.

Todos los diagramas desarrollados en la fase de diseño serán traducidos en un lenguaje máquina, en este caso Java, y que más adelante se compilará para generar el programa.

Para el desarrollo de la aplicación es necesario disponer previamente de un entorno Hadoop y Eclipse instalado y operativo. Existen multitud de máquinas virtuales compuestas por estos dos entornos, como *Cloudera* o una especialmente diseñada por *HortonWorks* para tal fin.

En nuestro desarrollo se ha querido iniciar desde cero por medio de tutoriales existentes en la red para instalar los componentes necesarios para que los dos entornos funcionen correctamente.

Dado que la implementación del código se realiza en Java, no se quiere entrar en profundidad en la instalación de la máquina virtual, por lo que se dejará una url de referencia donde se explica paso a paso como instalarla para un correcto funcionamiento de la misma.

Cuando el sistema operativo está completo con Hadoop y Eclipse solo será necesario instalar el entorno gráfico Java Swing, ubicado en los complementos que otorga la descarga de Eclipse. Mediante *Eclipse Marketplace* de la herramienta Eclipse encontraremos este complemento que instalaremos para poder comenzar con la implementación del código.

Ya se comentó con anterioridad el aspecto de este interfaz, por lo que no se incidirá más en este aspecto.

La estructura de Java Swing no deja mucha imaginación en la implementación de código. Recordemos que al ser un drag and drop todos los componentes funcionales y visuales de la aplicación implementan su propio código ya establecido y que solo deja cambiar aspectos visuales de los mismos. El funcionamiento corre a cargo del programador y es ahí donde se incide en esta sección.

Para hacerse una idea de cómo se ha estructurado el código, que mejor que una captura de la estructura de ficheros en Elipse:

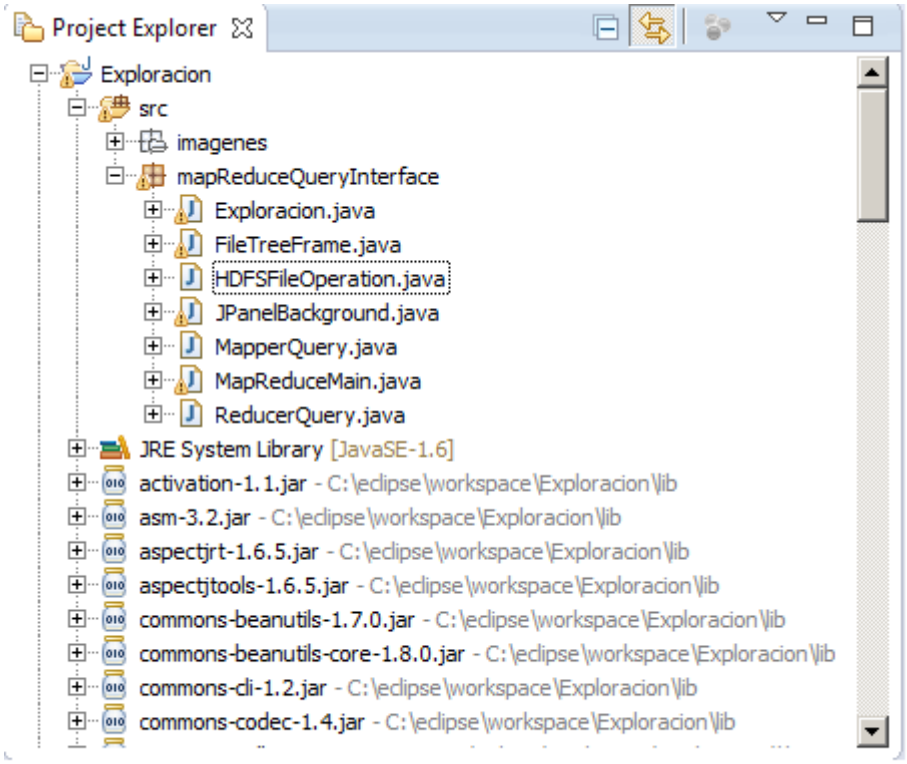


Ilustración 29: Esquema Eclipse de la aplicación

El proyecto creado tendrá el nombre de *Exploración* y en él se han generado las diferentes clases para la implementación del mismo. Principalmente destacar que la clase Exploracion.java es la principal, la cual lleva implícito todo el código relativo al diseño y carga de componentes al igual que el main y las funciones creadas según se requiera para la aplicación.

Para entender el esquema de clases es necesario generar un diagrama de clases que expone en su totalidad la arquitectura tomada para implementación de clases. Este diagrama ha sido generado mediante el plugin *ObjectAid UML[34]* desarrollado para Eclipse, herramienta capaz de generar clases UML y diagramas de secuencia a partir de código generad en Java, es decir, mediante ingeniería inversa.

La siguiente captura refleja las clases que se han desarrollado para la aplicación:

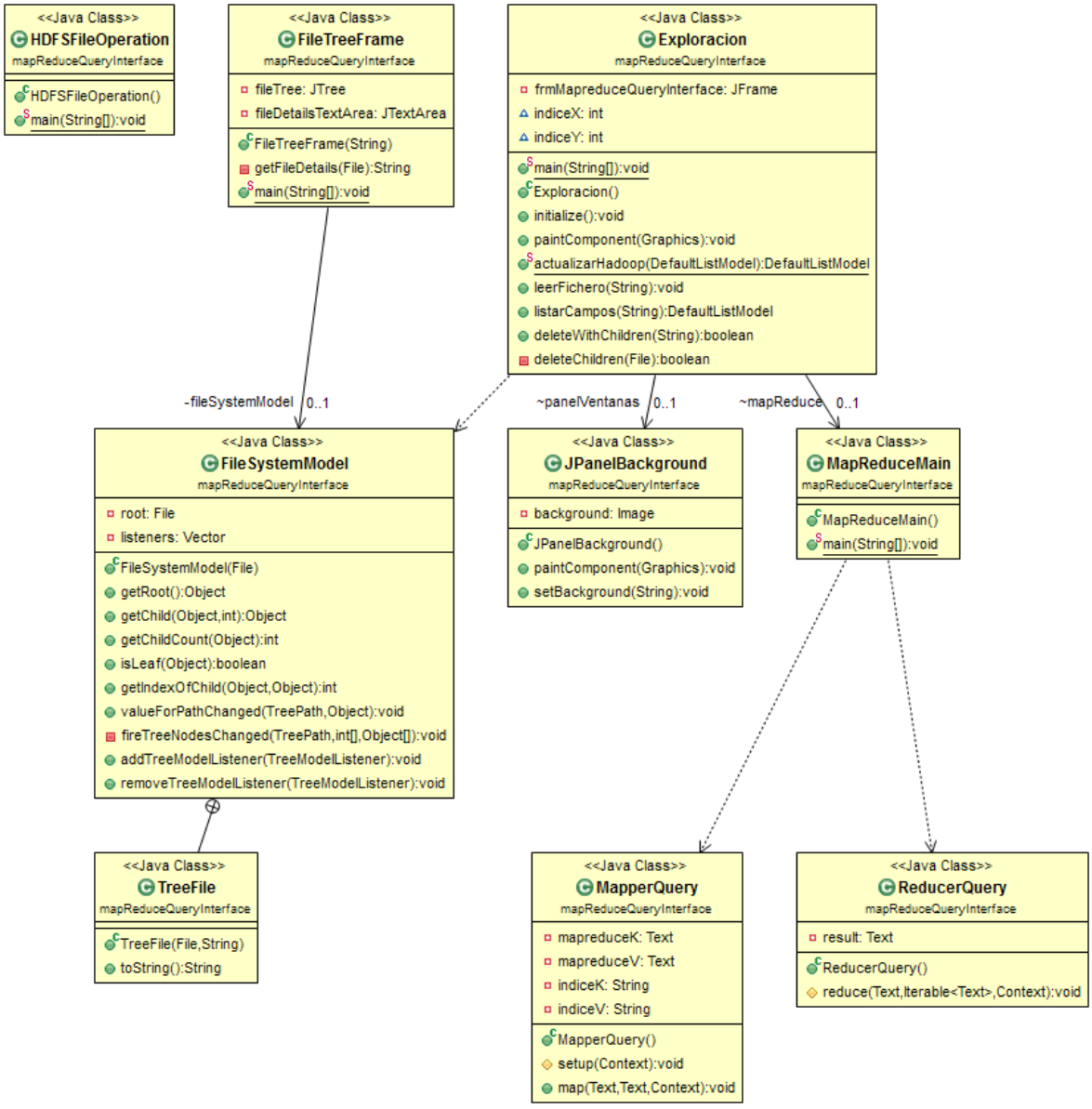


Ilustración 30: Diagrama de clases

Muchas de las clases generadas en el diagrama corresponden al framework Java Swing, el cual genera estas clases para poder tratar los componentes con *drag and drop*. Dado que este tipo de

clases no están generadas por uno mismo y no aportan relevancia en la implementación del código, se ha decidido no incluirla en el siguiente listado de clases que se expone a continuación. Es decir, sólo se expondrán aquellas clases que se han creado para la aplicación por uno mismo sin tener en cuenta las generadas automáticamente por el framework.

- *Clase Exploracion.java*

Como se ha comentado, esta clase es el corazón de la aplicación. Todos los componentes desarrollados y sus funcionalidades estarán presentes en esta clase.

Java Swing permite posicionar toda clase de componentes donde uno desee, se le aplicarán propiedades predefinidas para luego cambiarlas como uno quiera. Entre las más utilizadas dentro de la aplicación destacamos los JList, JButton, JLabels u OptionsButtons. Todos estos componentes tienen sus propios eventos para lanzar, ya sean arrastrando el puntero, seleccionándolo, realizando un focus y un sinfín de eventos que se nos ocurran tendrá Java Swing.

Es decir, la parte más complicada de un entorno visual casi es instantánea con este framework pudiendo dar multitud de posibilidades al desarrollador dejando la parte funcional aparte, que es la requiere mayor carga de trabajo.

La estructura de esta clase es bastante compleja y no hay que perderse en ella. Uno de los defectos de Java Swing es que se estructura por capas, es decir cada vez que se inserta un nuevo componente sobre una capa ya definida, todos los componentes pueden cambiarse al mismo tiempo si hablamos de mismas propiedades, como puede ser el tamaño, estilo o diseño. Aunque parezca un punto a favor, sólo será realmente óptimo cuando se implemente el diseño totalmente estructurado, empezando desde las capas más inferiores a las superiores siempre que se tenga en cuenta que un nuevo componente en una capa intermedia cuando ya se tenga una superior puede afectar al funcionamiento o diseño de la aplicación.

Los métodos generados en esta clase son los que se presentan con un recuadro rojo en la siguiente captura de pantalla:

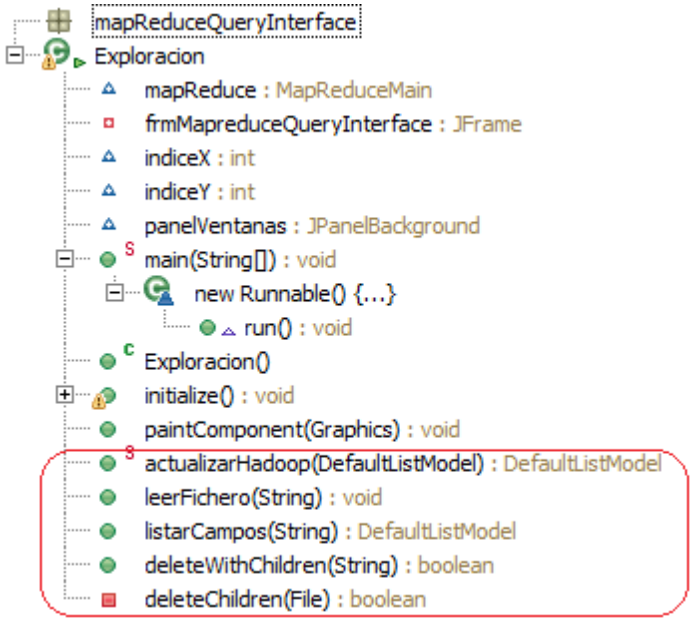


Ilustración 31: Esquema métodos de la aplicación en Eclipse

Se ha querido reducir al máximo la creación de métodos y clases para esta aplicación, de manera que no se tenga que explicar con gran profundidad el funcionamiento de los mismos y así favorecer a la creación de futuros métodos y clases de una manera sencilla y sin pérdidas para otro desarrollador.

Por ello, observamos que disponemos de cuatro métodos claramente diferenciados dentro de esta clase, cada uno de ellos creados para un fin en concreto.

- **actualizarHadoop**: Método cuya funcionalidad es refrescar el listado principal de Hadoop. Para realizar este método es necesario disponer de la información previa a la creación y ajustes del nodo Hadoop. Existen 2 ficheros que mantienen guardado la configuración previa del estado nodo que se ha creado cuando se inicia el programa. El primer fichero, core-site.xml, almacena la información del core Hadoop mientras que el fichero hdfs-site.xml contiene el listado del explorador HDFS que se abre en la sesión.

  La librería org.apache.hadoop contiene todos los métodos necesarios para almacenar esta información y poder trabajar con ella, por lo que no será necesario crear ningún método nuevo para poder acceder a toda la información referente a Hadoop. Se debe recoger el path del sistema Hadoop para poder actualizar los valores nuevos, ya que recordemos que es posible añadir nuevos ficheros en el listado Hadoop de la pantalla principal y además borrarlos en cualquier momento, por lo que la información reflejada debe estar actualizada en cada cambio que se realice.

- **leerFichero**: En este método también se debe recoger la información del core y del sistema de ficheros Hadoop para su funcionamiento. Mediante el nombre del fichero recibido como parámetro de entrada se puede conocer su ubicación y permitir el

acceso a Hadoop para su lectura. Dado que es otro sistema de ficheros diferente a Windows o Linux, la librería Hadoop también está preparada para realizar lecturas de ficheros mediante diferentes métodos que devuelven una lectura total de todo el contenido y estructura.

Este método tiene como finalidad mostrar el contenido del fichero seleccionado en la pantalla File Search, en un componente jframe creado para tal fin. Este componente visualizará el contenido del fichero, pero solo las 100 primeras líneas ya que es posible saturar al sistema si el contenido del fichero es muy extenso (se puede trabajar con ficheros de terabytes de datos, por lo que es necesario esa limitación).

- **listarCampos**: Al igual que los dos métodos anteriores, listarCampos también necesita la información de la configuración Hadoop para poder acceder al directorio con los ficheros HDFS. Tiene como objetivo mostrar cada campo referenciado en el fichero seleccionado.
  Si recordamos, existe una estructura clara para los ficheros .csv. Estos ficheros disponen de una cabecera estructurada en campos diferenciados por el carácter ";", por lo que es posible recoger esa información para mostrarla por pantalla.
  Éste método se utiliza en todas las listas Hadoop del programa, como en la pantalla principal o la ventana MapReduce.

  Su funcionamiento es sencillo: Recibe como parámetro de entrada el nombre del fichero seleccionado y luego la configuración de Hadoop permite acceder a su información de cabecera por un método ya creado para tal fin. Simplemente se debe recoger esa información para mostrarla en las textBox de la pantalla principal o en el proceso de selección de fichero para realizar el paradigma MapReduce.
  Los campos procesados servirán para entender la estructura del fichero o poder elegir cuáles serán los campos con los cuales se quiere procesar la información en MapReduce

- **deleteChilden** y **deleteWithChilden**: Cuando se generan resultados con el paradigma MapReduce se crea un fichero nuevo cada vez. Hadoop no contempla la posibilidad de reemplazo del fichero con nueva información, es decir, no se puede sobrescribir el propio fichero de resultados por lo que será necesario eliminarlo cada vez que se ejecute el paradigma.
  Este directorio de salida se encuentra en el sistema de ficheros Windows, no en Hadoop por lo que será necesario controlar todas las carpetas que crea Hadoop cada vez que se ejecuta. Se eliminó la carpeta por defecto preestablecida y se ha creado una nueva en C:\hdp-folders\.
  La estructura de ficheros generada por Hadoop se muestra en la siguiente captura, donde el fichero delimitado en rojo es el resultado del MapReduce, ubicado dentro de la carpeta output que crea el framework:

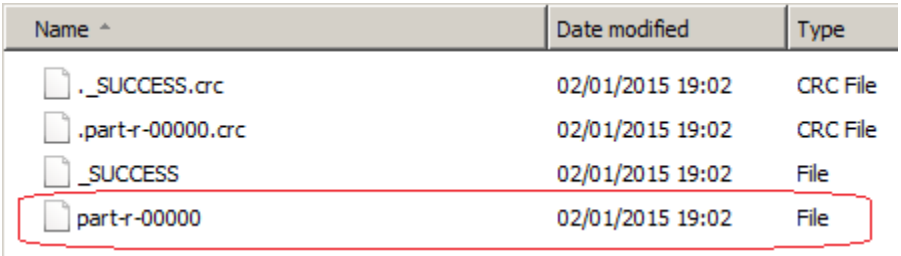


Ilustración 32: Captura pantalla Directorio Output

Cada vez que el paradigma sea lanzado, será necesario eliminar la carpeta output, ya que si no se lanzará una excepción de que ya existe tal carpeta. La finalidad del método será eliminar toda la carpeta output para renovar su contenido.

- *Clase FileTreeFrame.java*

Esta clase se genera automáticamente cuando se incluye el componente JTree. Este componente tiene como finalidad mostrar un árbol de ficheros en el propio contenedor, incluyendo carpetas y subcarpetas. Podemos decir que es igual que el componente explorador de cualquier Windows, donde podemos ver todos los ficheros y las carpetas padre que los contienen. Dado que para mostrar el listado y su estructura se necesita una programación acorde al componente, Java Swing otorga al programador su propia clase para un funcionamiento perfecto.

No se entrará más en detalle sobre esta clase ya que al ser autogenerada se entiende que no es importante para la implementación del código explicar su funcionamiento al completo. Sólo será necesario pasarle el path para comenzar con el listado de ficheros a partir de esa ubicación.

- *Clase HDFSFileOperation.java*

Esta clase contiene todos los métodos referentes al reinicio de ficheros Hadoop. Se ha reducido su contenido del original, ya que muchos métodos no eran necesarios.

Además se han cambiado otros para adaptarse a nuestra aplicación. Es importante saber que esta clase sólo dispone del método main() y cuya funcionalidad se describe a continuación:

- **Método main():** Comienza generando el directorio principal para los ficheros HDFS en la ubicación del entorno de trabajo de Hadoop en la carpeta *MyDataFolder.* Ésta carpeta almacenará todas las cargas de ficheros realizadas por el usuario, al igual que la eliminación de los mismos si el usuario lo solicita.

Al tratarse de un reinicio de la configuración previa de Hadoop, se elimina el directorio input de ficheros, se crea un fichero en la ubicación Hadoop y se escribe sobre el mismo.

La finalidad es comprobar si el acceso a HDFS es correcto y la mejor forma es probándolo la primera vez para así no tener futuros problemas.

Si es la primera vez que se inicia la aplicación en el ordenador, se lanzará esta clase para estructurar todo el sistema necesario para manejar el programa.

- *Clase JPanelBackground.java*

Uno de los inconvenientes de los componentes JFrame en Java Swing es que no permiten incluir imágenes externas al propio componente, para dotar a la interfaz de un aspecto más cuidado. Es por ello que se creó esta clase, para permitir poner imágenes background sobre un JFrame.

Mediante las propiedades width y height es posible ajustar la imagen con los pixeles que deseemos, mientras que el método drawImage de la librería java.awt permite recibir una imagen de entrada para pintarla sobre el componente.

Los constructores de la clase han sido modificados para tratar únicamente con una carpeta específica, donde se encuentran todas las imágenes que se muestran a lo largo de todas las ventanas de la aplicación.

- *Clase MapReduceMain.java*

Es la clase núcleo del paradigma MapReduce. Mediante la misma se enlazará el fichero seleccionado con las otras dos siguientes clases, MapperQuery y ReducerQuery.

Recibe como parámetros de entrada un listado con los ficheros seleccionados, un listado con las keys y otro listado con los values.

El método MapReduce distribuye las tareas a través de múltiples nodos y cada nodo proceso los datos siempre que sea posible. Además permite paralelizar y distribuir de manera casi automática, obteniendo una tolerancia a fallos realmente baja.

```
Job job = new Job(conf, "mapReduce");
job.setJarByClass(MapReduceMain.class);
job.setMapperClass(MapperQuery.class);
job.setReducerClass(ReducerQuery.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setInputFormatClass(KeyValueTextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
```

Ilustración 33: Jobs en MapReduce utilizados en la aplicación

Los denominados Jobs o trabajos en MapReduce son necesarios para atribuir configuraciones necesaria para que las clases puedan interactuar entre ellas, otorgando las clases mapper y reduce por ejemplo.

Antes de hacer referencia a las siguientes clases será necesario incluir todos los parámetros de entrada al método main y la propia clase será la encargada de lanzar el paradigma.

- *Clase MapperQuery.java*

La primera fase de MapReduce es la del Mapper. Estos mappers deben ejecutarse en nodos que mantienen una posición de los datos locales para evitar que se sature la red. La velocidad de lectura en Hadoop se mantiene alta gracias a que estos Mappers se ejecutan en paralelo, donde cada uno de ellos se procesa con un parte de los datos de entrada.

Su estructura es en forma de pares clave-valor como ya se ha comentado con anterioridad, donde cada key y value entregado generará una lista de la siguiente forma:

map(clave_entrada,valor_entrada)->lista (clave, valor)

El funcionamiento consiste en almacenar todos los datos relativos a las keys y values seleccionados por el usuario del propio fichero a tratar, para generar un listado de todos los datos que cumplan esa condición. Los tokens almacenados en la lectura del fichero se irán almacenando en un listado global que servirá de entrada para el siguiente proceso del MapReduce.

Con las pruebas seguramente se vea más claro todo el proceso

- *Clase ReducerQuery.java*

Una vez se pasa el proceso de Mapper, esta clase comienza la reducción de los valores que se repiten. Es decir, es un clase diseñada para recibir el listado general de clave-valor con los datos que cumplan la condición pedida para más adelante reducir todos aquellos datos que se repitan.

Todos los valores intermedios para una misma clave intermedia obtenida son combinados juntos en una lista. La lista es dada al reductor:

- Pueden existir más de un reductor al mismo tiempo
- Todos los valores asociados con una clave intermedia en particular van al mismo reductor
- Tanto las claves intermedias como su lista de valores son pasados al reductor ordenados por la clave
- Se conoce como *Shuffle and Sort*

El resultado que se obtiene de esta fase es un 0 o más valores como pares Clave-valor, escritos en HDFS y sobre el fichero *part-r-00000* que se ubicará en C:\\hdp-folders\output\

Para hacernos una idea de la fase MapReduce se expone la siguiente ilustración sobre un ejemplo de clustering teórico donde varias entradas disponen de distintas listas clave-valor para generar los resultados:

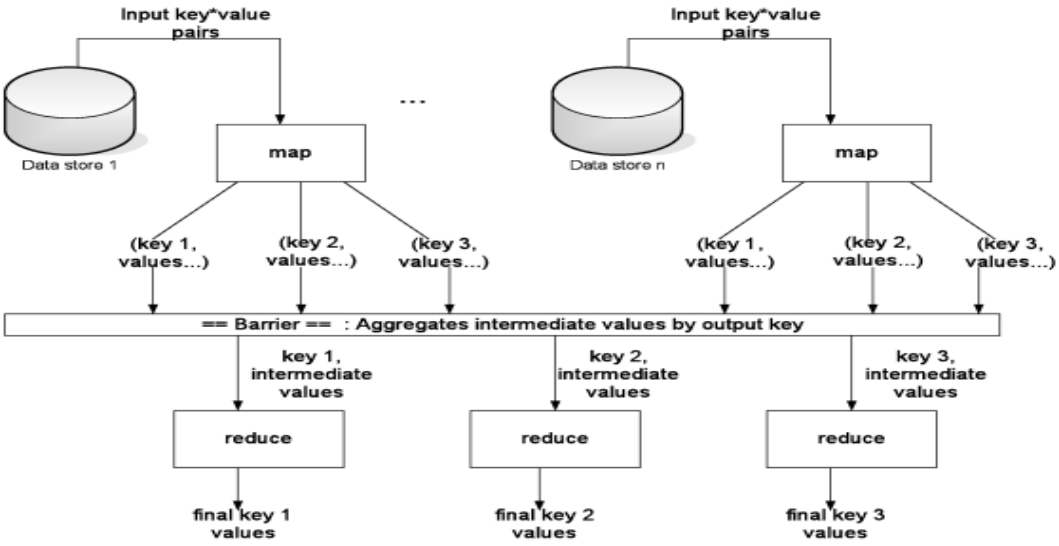


Ilustración 34: Proceso MapReduce con un ejemplo [35]

4.2…4 Pruebas

El objetivo de esta fase es bastante claro, comprobar que no se produzca ningún tipo de error en cualquiera de las anteriores fases, implicándonos más en la fase de implementación del código. Deberán probarse todos los módulos del sistema.

Se realizarán pruebas unitarias sobre la aplicación, probando cada módulo del sistema en su totalidad para comprobar que la funcionalidad del sistema es la correcta.

Para ello, se ha diseñado una tabla donde se recoge toda la información de la prueba a realizar. Dado que no es necesario incluir todas las pruebas, sólo se recogerá información útil de las mismas para tener una evaluación global del producto.

La realización de las pruebas se ha llevado a cabo con las siguientes especificaciones:

- Procesador Intel core i7
- 8GB de memoria RAM
- Disco duro 128GB SSD
- Windows Server 2008 R2
- Hadoop v2.0

En el caso de los ficheros a tratar se expone la siguiente tabla explicativa donde se recogen la información más importante de cada uno de ellos:

LISTADO DE FICHEROS PARA PRUEBAS		
NOMBRE FICHERO	**TAMAÑO**	**RECORDS**
InvertersPROD.csv	361KB	467
pantallasPROD-ID.csv	73,361 MB	29.665
09-2014.csv	534,98 MB	987.437
npidata_20050523-20141207.csv	5.573.121 MB	17.446.921

Tabla 35: Listado de ficheros utilizados para pruebas

- **InvertersPROD.csv:** Fichero específico de los productos Inverters dentro de una tienda online de Ebay donde se detallan diferentes atributos como nombre del producto, cantidad, precio, características, tipo de envío… Contiene 860 registros.

- **pantallasPROD-ID:** Fichero específico de los productos Pantallas dentro de una tienda online de Ebay donde se detallan diferentes atributos como nombre del producto, cantidad, precio, características, tipo de envío… Contiene 32.476 registros.

- **09-2014.csv:** Fichero con los datos del tráfico de la Comunidad de Madrid donde se especifican valores como el nombre de la calle, la fecha y hora, tiempo de retención si existiera, tramos, cruces… Consta de 202.664 registros.

- **npidata_20050523-20141207.csv:** Fichero con los datos suministrados por la agencia nacional de Estados Unidos del identificador estándar. Entre los campos más relevantes que pueden encontrarse están Entity Type Code, Employer Identification Number, Provider Business Mailing Address Country Code, Provider First Name o Provider Other Credential Text. Dispone de 37 campos como cabecera y más de 1 millón de datos comprendidos entre 23-05-2005 y 07-12-2014.

Es importante tener en cuenta que cada fichero está estructura de manera diferente, es decir, cada uno dispone de un número de campos diferentes lo cual repercute al paradigma MapReduce si utilizamos más o menos campos para el listado clave-valor.

Otro punto a tener en cuenta es el número de registros que se han registrado en el resultado final. Este dato se recoge en la columna Records y especifica el número de líneas del fichero generado.

Ya se sabe, a mayor número de registros, mayor es el tiempo empleado por el paradigma para recorrerlo en su totalidad.

Las pruebas se dividen según los módulos del sistema, por lo que cada uno de ellos tendrá una prueba específica según el funcionamiento a tratar.

- **PRUEBAS MODULO CARGAR FICHERO A HDFS**

  El objetivo de esta prueba es comprobar que los ficheros seleccionados en local se cargan en Hadoop para su posterior utilización. El tiempo que se utiliza para esta prueba viene definido en la columna “tiempo empleado” y es comprensible que a mayor tamaño del fichero, mayor será su tiempo de carga en Hadoop.

CARGAR FICHERO LOCAL A HDFS		
NOMBRE FICHERO	**TAMAÑO**	**TIEMPO EMPLEADO**
InvertersPROD.csv	361KB	0,6 segundos
pantallasPROD-ID	73,361 MB	1,3 segundos
09-2014	534,98 MB	9,2 segundos
npidata_20050523-20141207	5.573.121 MB	5,30 minutos

Tabla 36: Pruebas modulo cargar fichero local a HDFS

- **PRUEBAS MODULO LISTAR CAMPOS FICHERO**

  Estas pruebas tienen como objetivo comprobar que el fichero ya cargado en Hadoop se lista en la pantalla principal con todos los campos de los cuales consta el propio fichero.
  Para realizar esta prueba es obligatorio que el fichero ya se encuentre en Hadoop y aparezca en el listado de ficheros Hadoop del programa. Una vez se cumple lo anterior, simplemente debe clicarse sobre el mismo para que aparezcan todos los campos del fichero en un textbox.
  Se entiende que a mayor número de campos del fichero, mayor será su tiempo empleado para tal proceso. En la siguiente tabla se detallan los resultados:

LISTAR CAMPOS FICHERO HDFS			
NOMBRE FICHERO	**TAMAÑO**	**NUMERO DE CAMPOS**	**TIEMPO EMPLEADO**
InvertersPROD.csv	361KB	15	0,9 segundos
pantallasPROD-ID.csv	73,361 MB	15	0,9 segundos
09-2014.csv	534,98 MB	9	1,3 segundos
npidata_20050523-20141207.csv	5.573.121 MB	37	2,4 segundos

Tabla 37: Pruebas módulo listar campos fichero

- **PRUEBAS MODULO MAPREDUCE**

  Las pruebas que se realizan en este apartado tienen como objetivo comprobar el correcto funcionamiento del paradigma MapReduce sobre todos los ficheros anteriores.
  Para realizar tales pruebas es necesario que los ficheros estén cargados en Hadoop y se pueda acceder a ellos en la pantalla de MapReduce. Dado que cada prueba es totalmente diferente a la anterior, se ha querido abarcar diferentes keys y values de cada fichero para que los resultados sean lo más diferentes posibles dentro de una misma prueba sobre un fichero.
  El tiempo de respuesta para generar el fichero de resultados no depende en su totalidad del número de campos seleccionados tanto en keys como el values, sino de todos los datos que cumplen el mismo cluster. Cuantos más clusters generados por el paradigma, mayor será el tiempo empleado para resolverlo.
  La columna “records” muestra el número de registros generados en el fichero de respuesta del paradigma, por lo que a mayor número de estos, mayor será el tiempo empleado.

  Todo esto puede verse en la tabla que se muestra a continuación:

MAPREDUCE						
NOMBRE FICHERO	**TAMAÑO**	**NUMERO DE CAMPOS**	**NUMERO DE KEYS**	**NUMERO DE VALUES**	**TIEMPO EMPLEADO**	**RECORDS**
InvertersPROD.csv	361KB	15	1	2	2,5 segundos	467
InvertersPROD.csv	361KB	15	5	3	2,9 segundos	498
pantallasPROD-ID.csv	73,361 MB	15	3	4	7,7 segundos	29.665
pantallasPROD-ID.csv	73,361 MB	15	5	3	4,6 segundos	27.419
09-2014.csv	534,98 MB	9	1	1	1,46 minutos	80.002
09-2014.csv	534,98 MB	9	2	2	11,52 minutos	114.572
09-2014.csv	534,98 MB	9	3	3	19,30 minutos	487.996
npidata_20050523-20141207.csv	5.573.121 MB	37	1	1	9,37 minutos	1.324.113
npidata_20050523-20141207.csv	5.573.121 MB	37	2	3	1 hora y 35 minutos	1.435.784
5 ficheros estructura invertersPROD	1,5 MB Total ficheros	15	3	4	1,6 segundos	6.931
5 ficheros estructura invertersPROD	1,5 MB Total ficheros	15	8	4	3,6 segundos	7.028
15 ficheros estructura invertersPROD	5 MB Total ficheros	15	5	5	12,2 segundos	70.998
15 ficheros estructura invertersPROD	5 MB Total ficheros	15	8	8	11 segundos	67.754

Tabla 38: Pruebas módulo MapReduce

5 Evaluación

En este apartado se deben exponer los procesos que se han realizado para comprobar el correcto funcionamiento de la aplicación desarrollada.

La forma de evaluación se desarrolla en dos fases: la primera consta de seleccionar las diferentes funcionalidades de la aplicación y ejecutarlos para posteriormente analizarlos en la segunda fase.

Se seguirá la misma estructura de evaluación que en las pruebas del apartado 4:

- CARGAR FICHERO LOCAL A HDFS
- LISTAR CAMPOS FICHERO HDFS
- MAPREDUCE

En los siguientes apartados se detallarán la evaluación que se ha llevado a cabo y los resultados obtenidos de la misma.

Cabe destacar que la aplicación ha sido realizada con pruebas unitarias, funcionales y de rendimiento.

5.1 Proceso de evaluación

El proceso de evaluación corresponde a la primera parte de la evaluación del sistema desarrollado. Se debe escoger y diseñar el método de evaluación para la aplicación para luego poder ejecutar dicho método.

➢ Diseño de la evaluación

Para esta aplicación en concreto se han generado un conjunto de escenarios que puedan reflejar el funcionamiento real de la aplicación por un usuario. Dichos escenarios deben mostrar las tareas a realizar y además deben reflejar o verificar todos los requisitos del sistema.

Los escenarios están relacionados con las pruebas, por lo que se dispondrá de varios escenarios distintos. Cada uno de ellos debe reflejar todos los casos de uso posibles que puedan tratarse en el sistema, por lo que serán muy completos en ese aspecto.

Para realizar los escenarios se utilizará la siguiente plantilla:

Nombre del escenario	Identificador
Descripción	
Tareas a realizar	
Verificación	

Tabla 39: Plantilla escenarios

Donde cada uno de los campos especifica lo siguiente:

- **Nombre del escenario:** Descripción breve del escenario a tratar
- **Identificador:** Caracteriza al escenario en concreto unívocamente
- **Descripción:** Se especifica el escenario de forma extendida
- **Tareas a realizar:** Enumeración de cada una de las actividades que se han ejecutado para reproducir el escenario de evaluación
- **Verificación:** Son los resultados obtenidos después de haber ejecutado el escenario de evaluación

➢ Escenarios de evaluación

Como se ha comentado en el apartado anterior, los escenarios de evaluación se corresponden a las tres grandes funcionalidades del sistema (carga hdfs, lectura hdfs y mapreduce). Una vez se dispone de la plantilla para evaluar los escenarios, se procede con los mismos.

Nombre del escenario	**Carga y lectura de ficheros en HDFS** **Identificador** **ESCENARIO - 1**
Descripción	El usuario dispone de varios ficheros en local y desea trabajar con ellos en el sistema de ficheros HDFS para comprobar si la estructura del mismo es la correcta o debe cambiar el formato del mismo según lo que necesite. También quiere revisar parte del contenido del fichero para cerciorarse que es el fichero correcto. Al no disponer de toda la información necesaria a primera instancia en la ventana, decide ampliar el visionado del fichero para poder verlo al completo. Cuando el usuario esté conforme con el contenido del fichero, lo cargará para poder trabajar con él pero posteriormente se da cuenta que ese no era el fichero que necesitaba, por lo que lo elimina del directorio Hadoop y carga el fichero correcto.
Tareas a realizar	• Seleccionar el botón File de la barra de herramientas superior de la aplicación. • En la nueva ventana abierta debe acceder a la ruta donde se encuentra el fichero que necesita • Mediante el botón View, el usuario comprueba que el contenido del fichero es el correcto, mostrándose solo una pequeña parte del mismo en la ventana. • El usuario selecciona Show me more… para poder comprobar el contenido total del fichero en un bloc de notas que se abre para la ocasión • Comprueba que es el correcto y cierra el bloc de notas • Procede a la carga del fichero seleccionado mediante el botón de LOAD

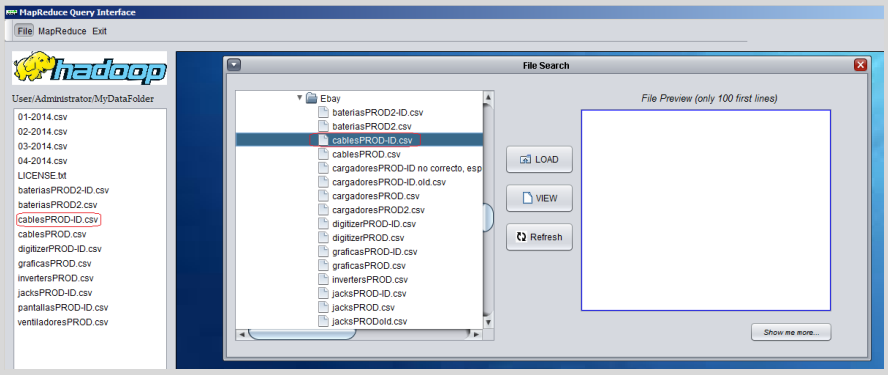
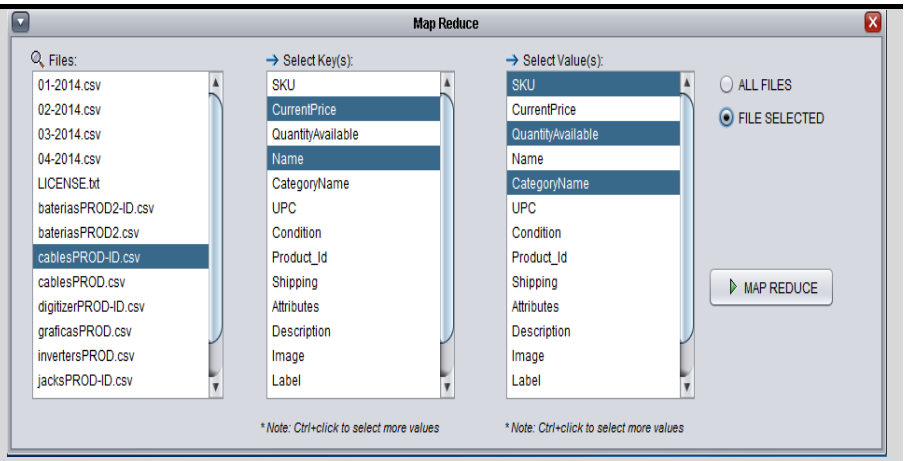
	• Cierra la ventana File Search • Comprueba que el fichero cargado se encuentra en el listado Hadoop de la ventana principal de la aplicación • El fichero cargado no es el que necesita por lo que clica sobre el botón Delete de la parte inferior del listado Hadoop • El fichero se elimina del listado y de su ubicación • El usuario realiza el mismo procedimiento de carga para el fichero correcto • Comprueba que se carga el fichero y es el correcto
Verificación	El usuario se encuentra en la ventana principal del programa y observa en el listado Hadoop, ubicado en la parte izquierda, que el fichero cargado está disponible para su selección y tratamiento. 

Tabla 40: Escenario - 1

Nombre del escenario	MapReduce sobre uno o varios ficheros HDFS	Identificador	ESCENARIO - 2
Descripción	El usuario dispone de un fichero ya cargado en HDFS y desea realizar MapReduce sobre el mismo. Para ello debe comprobar antes los campos disponibles del propio fichero y así estar seguro de que el fichero es el correcto. Una vez el usuario comprueba la veracidad del fichero, procede a realizar MapReduce. De entre los campos del fichero quiere relacionar el campo Name y CurrentPrice con los campos SKU, QuantityAvailable y CategoryName. Los dos primeros campos deben actuar como Keys mientras que los otros tres restantes deben ser Values.		

	Cuando la aplicación realiza todo el proceso, el usuario elige la opción de visualizar el resultado, pero los datos no concuerdan con lo que se suponía en un principio. Cae en la cuenta que debe seleccionar más ficheros que cumplan la misma estructura ya que todos los datos no se encuentran en un único fichero y es por ello que los resultados iniciales no son los esperados.
Tareas a realizar	• Comprobar la estructura del fichero cargado en HDFS mediante la acción doble click sobre el fichero en el listado principal de Hadoop, dentro de la ventana principal del programa • Aparece un textBox con la información referente a los campos del fichero • El usuario comprueba la estructura y lo da por válido • Selecciona el botón MapReduce de la barra de herramientas superior de la aplicación • Aparece una nueva ventana denominada MapReduce • El usuario escoge el fichero del primer listado disponible • Selecciona los campos Name y CurrentPrice del segundo listado Keys mediante la pulsación de ctrl+click • Selecciona los campos SKU, QuantityAvailable y Category del tercer listado Values mediante la pulsación de ctrl+click • Selecciona el radioButton File Selected para especificar que se quiere realizar el paradigma sobre ese único fichero • Pulsa sobre el botón Map Reduce de la ventana • Cuando finaliza el proceso, el usuario acepta visualizar el resultado en un bloc de notas • Comprueba que los resultados no son los esperados y realiza el mismo procedimiento pero cargando previamente los ficheros necesarios en HDFS • Comprueba que sólo están los ficheros necesarios en el listado Hadoop • En el proceso de Map Reduce en este caso selecciona All Files como radioButton para realizar el paradigma sobre todos los ficheros de la carpeta
Verificación	Una vez se lanza el paradigma MapReduce se debe comprobar el resultado final en el fichero generado. Debe aparecer en la primera fila los campos seleccionados para Keys y Values. Posteriormente deben aparecer los datos correspondientes de la forma lista(key/value). Se puede comprobar la veracidad del resultado consultando el fichero original y contrastando los datos.




```
10,99,CABLE DE VIDEO LCD FLEX NUEVO PARA DELL VOSTRO 1310,  |SKU-CF251310-1,54,CABLES DE PORTATIL,
100,01,CABLE DE VIDEO LCD FLEX NUEVO ACER ASPIRE 4741 G 4750 G 4551 G P/N: 50.4GW01.013,    |SKU-CF514741-78,48,CABLES DE PORTATIL,
100,01,CABLE VIDEO LCD FLEX NUEVO TOSHIBA SATELLITE A500 PANTALLAS LED P/N: DC02000UG00,    |SKU-CF405001-129,193,CABLES DE PORTATIL,
11,49,CABLE DE VIDEO LCD FLEX NUEVO DM3-1000 LED HPMH-B2695050G00001,   |SKU-CF200310-100,59,CABLES DE PORTATIL,
11,49,CABLE DE VIDEO LCD FLEX NUEVO PARA ACER ASPIRE 7720 7520 7220 7320 DC02000E100,   |SKU-CF517720-54,49,CABLES DE PORTATIL,
11,49,CABLE DE VIDEO LCD FLEX NUEVO PARA ACER aspire 7551 p/n: 50.4hn01.012,    |SKU-CF517551-74,50,CABLES DE PORTATIL,
11,49,CABLE DE VIDEO LCD FLEX NUEVO PARA TOSHIBA A200 A205 DC02000F900, |SKU-CF200205-20,51,CABLES DE PORTATIL,
118,99,CABLE DE VIDEO LCD FLEX NUEVO HP MINI CQ10 P/N: B2885050G00001,  |SKU-CF200101-148,55,CABLES DE PORTATIL,
120,99,CABLE DE VIDEO LCD FLEX NUEVO HP DV6 2000 PARA PANTALLA LED P/N: DD0UP8LC006,    |SKU-CF200006-99,51,CABLES DE PORTATIL,
120,99,CABLE DE VIDEO LCD FLEX NUEVO HP DV6 PARA PANTALLA LED P/N: DD0UP8LC006, |SKU-CF200006-138,38,CABLES DE PORTATIL,
121,29,CABLE DE VIDEO LCD FLEX NUEVO PARA TOSHIBA NB300 NB305 P/N:DC020007K00,  |SKU-CF403051-83,44,CABLES DE PORTATIL,
121,99,CABLE DE VIDEO LCD FLEX NUEVO TOSHIBA SATELLITE M800,    |SKU-CF400800-128,197,CABLES DE PORTATIL,
13,99,CABLE DE VIDEO LCD FLEX C710 C720 C730 C732 C722 C740 C750 C760 C770 C780 C790,   |SKU-CF200700-131,199,CABLES DE PORTATIL,
13,99,CABLE DE VIDEO LCD FLEX NUEVO COMPAQ C700 G7000 P/N: DC02000GY00, |SKU-CF200700-130,142,CABLES DE PORTATIL,
13,99,CABLE VIDEO LCD C715 C725 C735 C745 C755 C765 C775 C711 C715 C785 DC02000GY00,    |SKU-CF200700-132,185,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO ACER 5738 5536 5542 5236 CON CONEXIÓN A CAMARA WEB, |SKU-CF515536-136,50,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9000 DV9500,   |SKU-CF209000-62,519,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9310 DV9320 DV9330 DV9340 DV9350,  |SKU-CF209000-28,40,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9360 DV9370 DV9380 DV9390 DV9400,  |SKU-CF209000-60,40,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9410 DV9420 DV9430 DV9440 DV9450,  |SKU-CF209000-59,40,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9610 DV9620 DV9630 DV9640 DV9650,  |SKU-CF209000-27,40,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA HP COMPAQ DV9910 DV9920 DV9930 DV9940 DV9950,  |SKU-CF209000-112,40,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO LCD FLEX NUEVO PARA TOSHIBA P300 P/N: DD0BD3LC100, |SKU-CF400305-120,50,CABLES DE PORTATIL,
14,49,CABLE DE VIDEO P/N: P/N: FOXDD0AT9LC0011A HP DV9960 DV9970 DV9980 DV9990 DV9111,  |SKU-CF209000-81,507,CABLES DE PORTATIL,
14,49,CABLE DE VIdeo Flex P/N: FOXDD0AT9LC0011A HP DV9960 DV9970 DV9980 DV9990 DV9111,  |SKU-CF209000-127,498,CABLES DE PORTATIL,
```

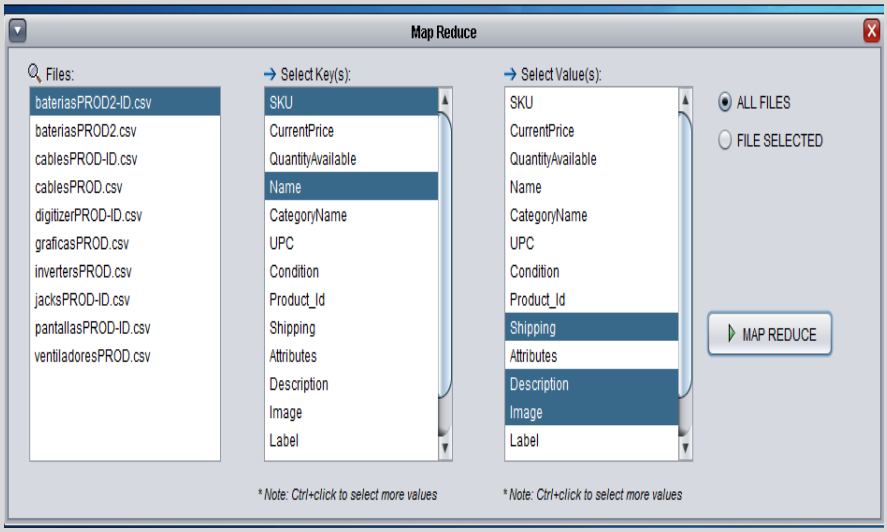
Tabla 41: Escenario - 2

Nombre del escenario	Carga de un directorio de ficheros con mismo formato y posterior MapReduce	Identificador	ESCENARIO - 3
Descripción	El usuario desea cargar en Hadoop todos los ficheros .csv dentro de un mismo directorio. Estos ficheros tienen la misma estructura, por lo que desea realizar MapReduce sobre los mismos en todo su conjunto. Para ello debe seleccionar la opción ALL FILES en la ventana MapReduce y así poder realizar el paradigma sobre todos los ficheros que tiene actualmente en Hadoop. Dado que es posible que existan ficheros anteriores en el listado Hadoop,		

	el usuario deberá eliminar previamente estos ficheros y cargar los nuevos.
Tareas a realizar	• Comprobar el listado izquierdo de Hadoop en la ventana principal del programa • Si existen ficheros anteriores deben ser borrados antes de realizar el MapReduce sobre todo el directorio Hadoop • Para ello debe seleccionar cada fichero del listado que quiere borrarse y se habilitará el botón DELETE en la parte inferior del propio listado. Debe realizarse esta acción hasta que todos los ficheros sean borrados del listado • Una vez concluido el apartado anterior, seleccionar el botón File de la barra de herramientas superior de la aplicación. • En la nueva ventana abierta debe acceder a la ruta donde se encuentra el directorio que necesita • Procede a la carga del fichero seleccionado mediante el botón de LOAD • El procedimiento de cargar el fichero debe realizarse con todos los ficheros que quieran ser cargados en la nueva ubicación de Hadoop. Se entiende que son todos los ficheros del mismo directorio al tratarse del mismo formato • Cierra la ventana File Search • Comprueba que el fichero cargado se encuentra en el listado Hadoop de la ventana principal de la aplicación • Selecciona el botón MapReduce de la barra de herramientas superior de la aplicación • Aparece una nueva ventana denominada MapReduce • El usuario escoge el fichero del primer listado disponible • Selecciona los campos SKU y Name del segundo listado Keys mediante la pulsación de ctrl+click • Selecciona los campos Shipping, Description y Image del tercer listado Values mediante la pulsación de ctrl+click • Selecciona el radioButton All files para especificar que se quiere realizar el paradigma sobre todos los ficheros ubicados en el directorio Hadoop. • Pulsa sobre el botón Map Reduce de la ventana • Cuando finaliza el proceso, el usuario acepta visualizar el resultado en un bloc de notas • Comprueba los resultados obtenidos
Verificación	El usuario comprueba que existen ficheros cargados con anterioridad en el directorio Hadoop. Procede a la eliminación de cada uno de ellos hasta dejar el listado Hadoop vacío. El siguiente paso es cargar todos los ficheros del directorio local a Hadoop mediante la ventana de File Search. Nuevamente comprueba que todos los ficheros cargados existen realmente en el listado Hadoop principal. Cuando selecciona el botón de MapReduce se abre la ventana y procede

a seleccionar un fichero Hadoop y sus respectivos Keys y Values según la elección del escenario a tratar. Antes de finalizar el proceso, el usuario selecciona All Files para realizar MapReduce sobre todos los ficheros del directorio Hadoop (deben disponer de la misma cabecera para poder tratarlos por igual).

Una vez finaliza el proceso MapReduce, el usuario decide visualizar el resultado final:




```
"SKU-A070VW04 7""-140",Pantalla para Portatil AU OPTRONICS A070VW04 A070AW04 MATE 40 PIN,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=1401
"SKU-A070VW04 7""-349",Pantalla para Portatil AU OPTRONICS A070VW04 A070AW04 MATE 40 PIN  V.0 V.1 V0,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jp
"SKU-B152EW01 15CON2 15,2""-18727","PANTALLA PORTATIL 15.2""  B152EW01  led COMPATIBLE SCREEN DISPLAY", |http://www.pantallasportatil.com/ebay/pantallas/normal.jp
"SKU-B152EW01 15CON2 15,2""-7638","PANTALLA PORTATIL 15.2""  B152EW01 V0  led COMPATIBLE SCREEN DISPLAY",   |http://www.pantallasportatil.com/ebay/pantallas/norma
SKU,Name,   |Shipping,Description,Image,|Shipping,Description,Image,|Shipping,Description,Image,|Shipping,Description,Image,|Shipping,Description,Image,|Shipping,
SKU-101-17181,PANTALLA  PORTATIL 10.1 WXVGA K000071910 LED, |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=17181|http://www.pantallasportatil.c
SKU-101B                    PP-16664,Nueva Pantalla PORTATIL Acer Aspire One A150 Series 10.1 LED ZG5, |http://www.pantallasportatil.com/ebay/pantallas/normal.jp
SKU-101B                    PP-23692,Pantalla PORTATIL Samsung  NC110 NP-NC110  10.1 LED CONECTOR ABAJO IZQUIERDA, |http://www.pantallasportatil.com/ebay/pantall
SKU-101B                    PP-25278,Pantalla screen Toshiba Netbook NB250 TODOS 10.1 MATE LED,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?pa
SKU-101B                    PP-27381,Pantalla screen Toshiba Netbook NB250  10.1 MATE LED, |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=273
SKU-101B                    PP-7731,Laptop Pantalla Compaq Mini CQ10-400 series 10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7731
SKU-101B                    PP-7734,Laptop Pantalla Fujitsu FMV-BIBLO LOOX M/D15  10.1 LED,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=
SKU-101B                    PP-7741,Laptop Pantalla HP Minibook CUD.M450-2H00S  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7741
SKU-101B                    PP-7742,Laptop Pantalla HP Minibook CUG.M450-2H00S  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7742
SKU-101B                    PP-7743,Laptop Pantalla HP MiniBook CUH.M455-2H00S  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7743
SKU-101B                    PP-7744,Laptop Pantalla Samsung N Series N210-JP01  10.1 Nueva LED,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?pa
SKU-101B                    PP-7745,Laptop Pantalla Samsung N Series NP-NP-135  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7745
SKU-101B                    PP-7746,Nueva Pantalla PORTATIL Acer Aspire One A150 Series 10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?pa
SKU-101B                    PP-7747,Nueva Pantalla PORTATIL Dell Latitude 2110  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7747
SKU-101B                    PP-7748,Nueva Pantalla PORTATIL Fujitsu FMV-BIBLO LOOX M/G30 (LMG30R)  10.1 LED,   |http://www.pantallasportatil.com/ebay/pantallas/n
SKU-101B                    PP-7749,Nueva Pantalla PORTATIL Gateway LT2118U  10.1 LED, |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7749|htt
SKU-101B                    PP-7750,Nueva Pantalla PORTATIL HP Mini 1100 series 10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7750
SKU-101B                    PP-7751,Nueva Pantalla PORTATIL HP Mini 2100 series 10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=7751
SKU-101B                    PP-8379,Pantalla portatil Nueva Samsung N Series NP-N220  10.1 LED,   |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?pa
SKU-101B                    PP-8380,Pantalla portatil Nueva Packard Bell EAV60  10.1 LED,  |http://www.pantallasportatil.com/ebay/pantallas/normal.jpg?param=8380
```

Observa que el resultado final concuerda con lo que se ha seleccionado en la aplicación y da como válido el fichero de respuesta.

Tabla 42: Escenario - 3

5.2 Análisis de resultados

Este apartado analiza los resultados que se han obtenido a raíz de las ejecuciones de los escenarios propuestos del apartado anterior.

El análisis de los mismos se realizará mediante una matriz de trazabilidad donde se relaciona cada requisito expuesto en el punto Especificación de requisitos con cada escenario de evaluación que se han presentado en el apartado anterior.

El objetivo de esta matriz es sencillamente corroborar que efectivamente se han cubierto todos y cada uno de los requisitos expuestos en los escenarios a tratar. Obviamente para que la verificación sea del todo correcta, todos los requisitos deberán estar relacionados con al menos un escenario de prueba.

En el caso de disponer de requisitos sin relacionar, se incluirán más escenarios de pruebas y si algún requisito cumple en más de un escenario, no tendrá repercusión alguna. Se entiende que un requisito puede tratarse en muchos escenarios ya que es posible que sea parte esencial de la aplicación.

Indicar que los requisitos reflejados en la matriz de trazabilidad son únicamente aquellos que definen la funcionalidad de la aplicación, es decir, los requisitos funcionales.

REQUISITO	ESCENARIO - 1	ESCENARIO - 2	ESCENARIO - 3
RF-1	X	X	X
RF-2	X		
RF-3	X		X
RF-4	X		X
RF-5	X		X
RF-6	X	X	X
RF-7	X	X	X
RF-8	X		
RF-9	X		
RF-10	X		
RF-11	X	X	X
RF-12	X	X	
RF-13	X		X
RF-14	X	X	X
RF-15		X	X
RF-16		X	X
RF-17	X		X
RF-18		X	X
RF-19		X	X
RF-20		X	X
RF-21		X	X
RF-22		X	X
RF-23		X	X
RF-24		X	X
RF-25		X	X

Tabla 43: Matriz trazabilidad requisitos-escenarios

6 Conclusión

El último apartado de esta memoria expone de manera resumida los objetivos que se han alcanzado, los problemas encontrados y las aportaciones recibidas a modo de conocimientos.

Se intenta aportar una síntesis del desarrollo de la aplicación, a la vez que se aportan las conclusiones obtenidas a lo largo de todas las fases del proyecto con un aporte de cara a futuras mejoras sobre la propia aplicación.

6.1 Conclusión

El rápido proceso de evolución que estamos sufriendo con las nuevas tecnologías está provocando que aspectos poco conocidos y sin aportaciones futuras estén tomando relevancia a día de hoy.

Uno de estos aspectos es el término Big Data. Hasta hace bien poco este término no se tenía en cuenta en prácticamente ninguna empresa tecnológica dado que la inversión inicial que tenía que aportarse era demasiado grande de cara a obtener beneficios a corto plazo. Por ello, aunque las empresas tenían bien presente los beneficios que podía aportar, pocas eran las que se lanzaban hacia una nueva tecnología desconocida.

La principal duda de las empresas a la hora de decantarse por esta tecnología es si realmente es interesante la propuesta Big Data y que beneficios aportaría. Bien pues la plataforma de código abierto Hadoop intenta dar el empujón definitivo a todas aquellas empresas que no han descubierto todavía el valor asociado a toda esa información generada diariamente.

Hadoop divide las tareas en Mapper y Reducer para manipular todos aquellos datos distribuidos a nodos de un clúster, logrando un paralelismo alto en el procesamiento de esa información. Al disponer de un sistema de ficheros propio, HDFS, se consigue dividir la información en bloques pequeños que se distribuyen a cada clúster y así tanto map como reduce pueden ejecutarse en pequeños subconjuntos otorgando gran escalabilidad en un amplio volumen de datos. En resumidas cuentas, aun teniendo una cantidad de información desproporcionada, Hadoop puede analizarla sin problemas, y lo más importante, con una velocidad nunca antes vista en una base de datos relacional.

El gran beneficio que aporta Hadoop es manejar cualquier tipo de información, de cualquier tipo de estructura sin importar nada el contenido del mismo, por lo que importar esos datos a una base de datos relacional se queda en el pasado. El futuro es Big Data y según avancemos hacia un futuro con una tecnología más presente en el día a día nos daremos cuenta que será una parte esencial para analizar toda esta información.

Si retomamos los objetivos del proyecto, se puede afirmar que se cumplen en su totalidad. Se ha estudiado el ámbito de las tecnologías Big Data y Hadoop, todo su ecosistema y cómo funcionan para otorgar soluciones a las empresas con una gran cantidad de información diaria.

La idea era crear una aplicación capaz de facilitar el intercambio de archivos entre local y Hadoop, permitiendo así su posterior lectura en el formato HDFS y proporcionando un interfaz capaz de realizar consultas MapReduce. Todos los apartados se han cumplido y se han evaluado con una tanda de pruebas para verificar si realmente cumplen la funcionalidad que se le pedía.

Se demuestra pues que la aplicación es funcionalmente operativa y cumple con todos los objetivos pactados antes de comenzar el TFG

6.2 Trabajos futuros

Una vez se entienden los beneficios que puede aportar la tecnología Big Data, uno se da cuenta de todo el abanico de posibilidades que se encuentra al alcance. Pero cuidado, no todo es beneficioso si los resultados obtenidos no son especialmente útiles para el consumidor o empresa que utilice esta tecnología.

Es por ello que debe tenerse presente el problema a tratar para así obtener los máximos beneficios de esta tecnología y esa una de las grandes dudas de las empresas a la hora de implantar el software para tratamiento de grandes volúmenes de información.

Con la aplicación desarrollada se intenta introducir al cliente a esa tecnología, que puede ver de primera mano los beneficios que podría aportarle de cara a su problema. Se trata de una aplicación sencilla y potente para el tratamiento de grandes ficheros sin disponer de una estructura claramente definida.

En este caso, se ha querido reducir el manejo de cualquier tipo de fichero debido al escaso margen de tiempo que se dispone para su desarrollo. Como ya se sabe, esta aplicación solo puede realizar tratamientos de ficheros con una extensión en concreto y que debe disponer de una cabecera inicial con una estructura definida.

De cara a trabajos futuros sobre la aplicación se intentará corregir esas carencias, aportando más variedad en el manejo de ficheros, con diferentes extensiones y a su vez que no sea necesario una estructura fija ya que eso delimita mucho a la hora de elegir un software u otro. La idea principal de mejora es proporcionar al usuario final un software rico en variedad de manejo de ficheros, sin límites, de tal manera que él mismo ponga sus límites en el tratamiento de información.

Otra de las aportaciones necesarias para el manejo de ficheros en HDFS es disponer de un árbol a modo de explorador de ficheros. Si recordamos el software presentado no deja margen para navegar por los directorios HDFS, sino que solo existe una ruta preestablecida para manejar ese tipo de ficheros.

Por lo tanto, una de las aportaciones futuras será proporcionar al usuario un manejo total sobre el sistema de ficheros para que él mismo pueda moverse por todo el disco duro y así poder seleccionar cualquier fichero que se encuentre en una ubicación diferente siempre que lo desee.

Es importante también disponer de escalabilidad en la aplicación. Las pruebas realizadas sobre el paradigma MapReduce reflejan un alto grado de lentitud en los resultados, cosa que no debería pasar con esta tecnología. El principal inconveniente de este software es su número de nodos,

muy reducido si tenemos en cuenta el número de ellos que puede trabajar en paralelo. Al tratarse de una única instancia Hadoop, solo se dispone de un nodo de datos, es decir un único clúster HDFS. La mejora implementada de cara a un futuro sería poder trabajar sobre múltiples máquinas con múltiples instancias Hadoop. Eso conlleva un aumento de la velocidad de procesamiento gracias también al replicado de datos a través de múltiples host lo que generaría un reequilibrado de datos o incluso copias automáticas de los bloques de ficheros

6.3 Problemas encontrados

Esta sección permite exponer todos los problemas que han ido surgiendo a lo largo de todo el desarrollo del proyecto. Se incluirán los problemas más importantes y los que hayan podido reflejar cambios en la estructura de la aplicación por diversos motivos.

- ### Diseño aplicación

Durante la fase de diseño se encontraron muchos problemas para desarrollar un interfaz amigable y potente. Se trataba de dar al usuario otra plataforma distinta a lo que existe actualmente en el mercado, por lo que se optó por incluir lectura de ficheros en una nueva ventana de diseño y un interfaz para MapReduce. Hubo que rediseñar la ventana de FileSearch para así dar preferencia al visionado parcial del fichero seleccionado

Cuando se diseñó la aplicación, no se sabía qué tipo de componentes introducir para dar un aspecto actual de una aplicación desarrollada para Windows. Una vez se utilizó Java Swing para el desarrollo de la aplicación, el diseño inicial tuvo que cambiarse por completo al encontrar nuevos componentes que cumplían a la perfección lo que se quería. Se podría decir que el aspecto del diseño ha variado unas cuantas veces antes de utilizar el framework de diseño de Java.

- ### Implementación código

El trabajo más costoso y con mayores problemas ha sido la implementación del código. Dado que nunca se había utilizado Hadoop, era bastante probable que surgieran dudas y problemas para implementarlo.

Aunque se hayan seguido tutoriales para instalar Hadoop con Eclipse, los problemas han sido muchos y muy complicados de solucionar. La máquina virtual que genera Hadoop para establecer el core y los nodos a utilizar no todas las veces se creaban bien, y hay que tener amplios conocimientos de esta estructura para entender los problemas que refleja la consola de Java.

Es por ello que la instalación de la plataforma ha producido muchos retrasos en las fechas establecidas para la finalización de la implementación de código. Se ha tenido que seguir varios tutoriales de instalación a la vez que iban surgiendo errores en la instalación, hasta que se ha utilizado una máquina virtual proporcionada por el tutor del TFG, que ya venía preestablecido con los programas necesarios para el desarrollo de la aplicación como podían ser Hadoop y Eclipse ya configurados para el entorno.

Problemas de código siempre surgen cuando se desarrolla una aplicación, simplemente cabe destacar el funcionamiento de los componentes. Desde un primer momento no se entiende el

funcionamiento de eventos y repintados de capas en Java Swing, pero con unas cuantas horas de implementación de código el concepto se va entendiendo y las posibilidades que otorga este framework resultan más que interesantes.

- ### Concepto MapReduce

Era necesario disponer de conocimientos previos al paradigma, para que luego en su desarrollo no resultase excesivamente complicado. Se ha tenido que revisar mucha documentación al respecto para entender el funcionamiento, que puede otorgar y como desarrollarlo.

Aun con toda la información recopilada y las horas dedicadas, el concepto es complicado de entender cuando durante toda la carrera se han utilizado bases de datos relacionales, por lo que inconscientemente intentamos plasmar esa nueva información en casos Big Data, pero es un tremendo error.

Este tipo de estructura se desarrolla en otro contexto diferente y hasta que no se entienda, no podrá conocerse el concepto MapReduce.

6.4 Opiniones personales

Una vez concluido el proyecto en su totalidad uno intenta ponerse en la situación inicial antes de comenzar con el TFG. Si echamos la vista atrás, sólo hablar de la tecnología Big Data era un auténtico embrollo que no se conocía o se tenía muy pocos conocimientos sobre el mismo.

Cuando se realizaron los primeros pasos no tenía otra salida que intentar entender el mundo del Big Data y cuáles son los beneficios que aportaría de cara al proyecto. Se realizaron múltiples estudios de esta tecnología en las empresas pioneras en el sector, cuales habían sido sus aportaciones y que factores habían sido los detonantes para utilizar Big Data.

Poco a poco uno se va enterando de cómo funciona hoy en día nuestro mundo y cae en la cuenta que no todo se rige por bases de datos estructuradas si tenemos la necesidad de guardar esa información. Uno sabe que la totalidad de las empresas necesitan una base de datos para almacenar la información y que hoy en día están aguantando muy bien sin necesidad de implementar nuevas mejoras.

Entonces, ¿Qué era necesario para una empresa decantarse por esta nueva tecnología si realmente les iba bien hasta ahora? Muy fácil, los datos. La información generada por una empresa en su día a día era descartada casi por completo ya que se decidió que no era lo suficientemente relevante como para almacenarla ya que era tal la cantidad de información que había que reducirla necesariamente.

Uno de los factores que me decidieron desarrollar esta aplicación era esta tecnología, ya que tarde o temprano sería prácticamente indispensable en las empresas grandes y pocas personas tienen la experiencia necesaria para ponerse al mando de un cambio radical en la forma de almacenar la información como es Big Data. Aunque existen muchas plataformas para trabajar con grandes volúmenes de datos, Hadoop me pareció la más correcta para el desarrollo de esta aplicación. Principalmente por ser gratuita y a la vez muy potente para el tratamiento de datos.

Además tengo que incluir la aportación del tutor en una de las asignaturas de motores de búsqueda, la cual fue el detonante para elegir este proyecto.

Aunque realmente tengo que decir que el ecosistema Hadoop-Eclipse está todavía muy limitado. Para un programador siempre es bueno disponer de una plataforma familiar como es Eclipse, la cual aporta grandes beneficios cuando se trata de desarrollar un producto. Pero desgraciadamente se ha perdido mucho tiempo en juntar ambas piezas, ya sea por novato o simplemente por el hecho de no conocer en su totalidad la manera de configurarlo correctamente, pero he de decir que con el tiempo suficiente y unos pocos conocimientos sobre la materia pueden hacerse cosas increíbles con Hadoop.

Solo se ha tenido la ocasión de desarrollar un único modelo de programación con Hadoop, el paradigma MapReduce. Pero solo por ello puedo decir que sus posibilidades son casi infinitas, casi cualquier problema que se quiera solucionar puede tratarse con ese paradigma. Eso sí, no todos reflejan los resultados que uno desea pero entiendo que a la larga se mejorará esta tecnología y pequeños baches en el camino se solucionarán en poco tiempo.

Todo el conocimiento que tengo de bases de datos relacionales ha sido indispensable para comprender Big Data y su funcionamiento ya que muy posiblemente desbanque a esa tecnología actual aportando beneficios para manejar grandes volúmenes que poco a poco van surgiendo.

Estoy realmente satisfecho con la tecnología empleada, me ha otorgado otro punto de vista diferente y a tratar grandes volúmenes de información que no creía posible analizar en un corto periodo de tiempo.

Atentos a la tecnología Big Data, sus beneficios son tan amplios que posiblemente encontremos soluciones a problemas que hasta la fecha son impensables de tratar, aportando mejoras beneficiosas para el día a día, como la medicina, la educación o la forma de obtener información

7 Bibliografía

[1] IBM, "IBM big data and information management," [Online]. Available: http://www-01.ibm.com/software/data/bigdata/

[2] Juan_Vidal, "Consideraciones procesos ETL en entornos Big Data: Caso Hadoop," 7 Abril 2014, [Online]. Available: http://www.dataprix.com/blog-it/big-data/consideraciones-procesos-etl-entornos-big-data-caso-hadoop

[3] Internality, "Mapa visual de la web 2.0," [Online]. Available: http://www.internality.com/web20/

[4] Francisco Javier Martínez Páez, "Apache Hadoop-HDFS," 15 Febrero 2014, [Online]. Available: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hadoopFS

[5] Microsoft, "Introducción a los sistemas de archivos FAT, HPFS y NTFS," 2 Julio 2013, [Online]. Available: http://support.microsoft.com/kb/100108/es

[6] Paradigma tecnológico, "Introducción a Apache Hadoop," 22 Diciembre 2011, [Online]. Available: http://es.slideshare.net/paradigmatecnologico/introduccion-apache-hadoop

[7] Joe Pasqual, "Schema on read vs Schema on write," 27 Febrero 2014, [Online]. Available: http://www.marklogic.com/blog/schema-on-read-vs-schema-on-write/

[8] Angel Acha Lizama, "Hadoop on Azure (II): Funciones MapReduce y procesamiento de datos de entrada," 18 Septiembre 2012, [Online]. Available: http://geeks.ms/blogs/ciin/archive/2012/09/18/hadoop-on-azure-ii-funciones-mapreduce-y-procesamiento-de-datos-de-entrada.aspx

[9] Ricardo Barranco Fragoso, "¿Qué es Big Data?," 18 Julio 2012, [Online]. Available: https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

[10] Sorprendida, "Big Data: de problemas de tecnología a solución de negocio," 14 Agosto 2014, [Online]. Available: http://www.madridgeekgirls.es/big-data-solucion-negocio/

[11] Stratebi, "Cursos Big Data Open Source," 11 Enero 2015, [Online]. Available: http://es.slideshare.net/zanorte/cursos-big-data-open-source

[12] Erick Camacho, "NoSQL la evolución de las bases de datos," 28 Mayo 2010, [Online]. Available: http://sg.com.mx/revista/42/nosql-la-evolucion-las-bases-datos#.VMI9aC5vCi0

[13] Esteban Saavedra, "Introducción a las bases de datos documentales en linux," 22 Noviembre 2012, [Online]. Available: http://es.slideshare.net/estebansaavedra/bases-de-datos-documentales-15300505

[14] Anthony R. Sotolongo León, "Bases de datos NOSQL orientadas a documentos," 26 Enero 2014, [Online]. Available: http://es.slideshare.net/asotolongo/bases-de-datos-nosql-orientadas-a-documentos

[15] O'Reilly Strata, "An introduction to Hadoop 2.0: Understanding the New Data Operating System," 3 Enero 2014, [Online]. Available: http://radar.oreilly.com/2014/01/an-introduction-to-hadoop-2-0-understanding-the-new-data-operating-system.html

[16] Rodrigo Jaramillo, "Presentacion Hadoop," 9 Diciembre 2013, [Online]. Available: https://prezi.com/0zlpwnwpvsb0/presentacion-hadoop/

[17] Luis Miguel Gracia, "Un poco de JSON-Schema," 10 Mayo 2014, [Online]. Available: http://about.sofia2.com/2014/04/10/un-poco-de-json-schema/

[18] Qlink, "Presentamos Qlink Sense: Libere su intuición", [Online]. Available: http://support.microsoft.com/kb/100108/es

[19] Datameer, "Webinars," [Online]. Available: http://www.datameer.com/learn/index.html

[20] Datameer, "Datameer 5.0: Optimized, intelligent, dynamic analytics," [Online]. Available: http://www.datameer.com/product/smart-execution.html

[21] Clasorra, "Tecnologías de Big Data: el ecosistema Hadoop," 30 Agosto 2013, [Online]. Available: http://blog.classora.com/2013/08/30/tecnologias-de-big-data-el-ecosistema-hadoop/

[22] Elena, "Configuración de Eclipse con Hadoop (local y pseudo-distribuido)," 21 Febrero 2013, [Online]. Available http://hadoopontheroad.blogspot.com.es/2013/02/configurando-eclipse-para-desarrollar.html

[23] Hadoop, "Hadoop Map/Reduce Tutorial," 12 Febreo 2008, [Online]. Available: http://hadoop.apache.org/docs/r0.18.3/mapred_tutorial.html#Mapper

[24] Wikipedia, "Diagrama de Gantt," 15 Enero 2015, [Online]. Available: http://es.wikipedia.org/wiki/Diagrama_de_Gantt

[25] Microsoft, "Realizar pruebas tempranas y frecuentes,", [Online]. Available: https://msdn.microsoft.com/es-es/library/vstudio/ee330950%28v=vs.110%29.aspx

[26] Jean Triquet, "Hadoop: Procesamiento de datos para Big Data," [Online]. Available: http://director-it.com/index.php?option=com_content&view=article&id=133:hadoop&catid=69:big-data&Itemid=116

[27] Nancy Riaño, "Ciclo de vida de un sistema," Mayo 2013, [Online]. Available: http://nancyriao.blogspot.com.es/

[28] James & Suzanne Robertson, "Volere: Plantilla de especificación de requisitos," 11 Febrero 2006, [Online]. Available: http://www.volere.co.uk/pdf%20files/template_es.pdf

[29] Wikipedia, "Caso de uso," 24 Diciembre 2014, [Online]. Available: http://es.wikipedia.org/wiki/Caso_de_uso

[30] Wikipedia, "Swing: Biblioteca gráfica," 31 Marzo 2014, [Online]. Available: http://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29

[31] Galo Latorre, "¿Qué es Swing?," 9 marzo 2009, [Online]. Available: http://gl-epn-programacion-ii.blogspot.com.es/2010/04/que-es-swing.html

[32] Ernesto Bastón Pantoja, “El patrón de diseño MVC y su implementación en java Swing," Diciembre 2014, [Online]. Available: http://www.academia.edu/5217432/El_patr%C3%B3n_de_dise%C3%B1o_Modelo-Vista-Controlador_MVC_y_su_implementaci%C3%B3n_en_Java_Swing

[33] Oscar Arrivi, "El patrón Modelo-Vista-Presentador (MVP) a examen," 3 Octubre 2010, [Online]. Available: http://theartoftheleftfoot.blogspot.com.es/2010/10/el-patron-modelo-vista-presentador-mvp.html

[34] Eduardo Terrasa, "Instalar ObjectAid UML plugin para Eclipse," [Online]. Available: http://www.edu4java.com/es/progbasica/progbasica16.html

[35] Francisco Javier Pulido, "Big Data & Hadoop (II) - MapReduce," [Online]. Available: http://www.franciscojavierpulido.com/2013/07/bigdata-hadoop-ii-mapreduce.html

7.1 Bibliografía complementaria

Esta bibliografía complementaria corresponde a los libros utilizados para obtener los conocimientos básicos de la temática tratada en el TFG. En ningún momento se utilizan en la memoria para nombrar citas al respecto.

[36] Robert D. Schneider, "Hadoop for Dummies," 2012. IBM

[37] Kevin T. Smith, Alexey Yakubovich y Boris Lublinsky, "Hadoop: Soluciones Big Data," Anaya-Multimedia. 2014

[38] Jason Venner, Sameer Wadkar y Madhu Siddalingaiah, "Pro Apache Hadoop," Apress (second edition). Septiembre 2014

[39] Tom White, "Hadoop: The Definitive Guide," O’Reilly. Mayo 2012

Anexo I. Control de versiones

Versión	Fecha de finalización	Descripción
0.1	15 septiembre 2014	Redacción: 1. Introducción
0.2	29 septiembre 2014	Redacción: 2. Estado de la cuestión 2.1 Tecnologías Big Data 2.2 Proyectos actuales
0.3	21 octubre 2014	Redacción: 3. Gestión de proyecto software 3.2 Plan de trabajo 3.5 Plan de pruebas
0.4	5 noviembre 2014	Revisión anteriores capítulos
0.5	9 noviembre 2014	Redacción: 4. Solución 4.1 Definición del problema 4.2 El proceso de desarrollo
0.6	14 noviembre 2014	Redacción: 4.2 El proceso de desarrollo : Análisis
0.7	21 noviembre 2014	Revisión Análisis
0.8	23 noviembre 2014	Redacción: 4.2 El proceso de desarrollo : Diseño
0.9	29 noviembre 2014	Revisión Diseño
1.0	3 diciembre 2014	Redacción: 4.2 El proceso de desarrollo : Implementación Diagrama de clases
1.1	4 diciembre 2014	Revisión general
1.5	9 diciembre 2014	Redacción: 4.2 El proceso de desarrollo : Implementación Diagrama de clases
2.0	26 diciembre 2014	Redacción: 4.2 El proceso de desarrollo : Pruebas Tablas pruebas con diferentes módulos
2.1	3 enero 2015	Revisión Implementación y pruebas

3.0	5 enero 2015	Redacción: 5. Evaluación 5.1 Proceso de evaluación Escenarios Matriz de trazabilidad req.-escenarios
3.1	12 enero 2015	Redacción: 6. Conclusión 6.1 Conclusión 6.2 Trabajos futuros
3.2	15 enero 2015	Redacción: 6. Conclusión 6.3 Problemas encontrados 6.4 Opiniones personales
3.3	15 enero 2015	Revisión general
4.0	23 enero 2015	Redacción: 7. Bibliografía
4.1	24 enero 2015	Revisión ilustraciones y tablas
4.2	25 enero 2015	Revisión bibliografía
4.3	26 enero 2015	Revisión general
4.4	8 febrero 2015	Revisión puntos 1 al 3
4.5	13 febrero 2015	Revisión puntos 4 y 5
4.6	18 febrero 2015	Revisión puntos 6 y bibliografía
4.7	19 febrero 2015	Revisión general

Anexo II. Seguimiento de proyecto fin de carrera

Este segundo anexo detalla la forma de seguimiento que se ha llevado a cabo en el trabajo fin de grado. Para realizarlo, es necesario en primera instancia establecer la forma con la cual se ha desarrollado el propio proyecto para posteriormente establecer una planificación desde el comienzo del trabajo fin de grado hasta su finalización.
A lo largo de esta planificación se detallan las tareas establecidas y los periodos de tiempo asociados para cada tarea en su realización.

Forma de seguimiento

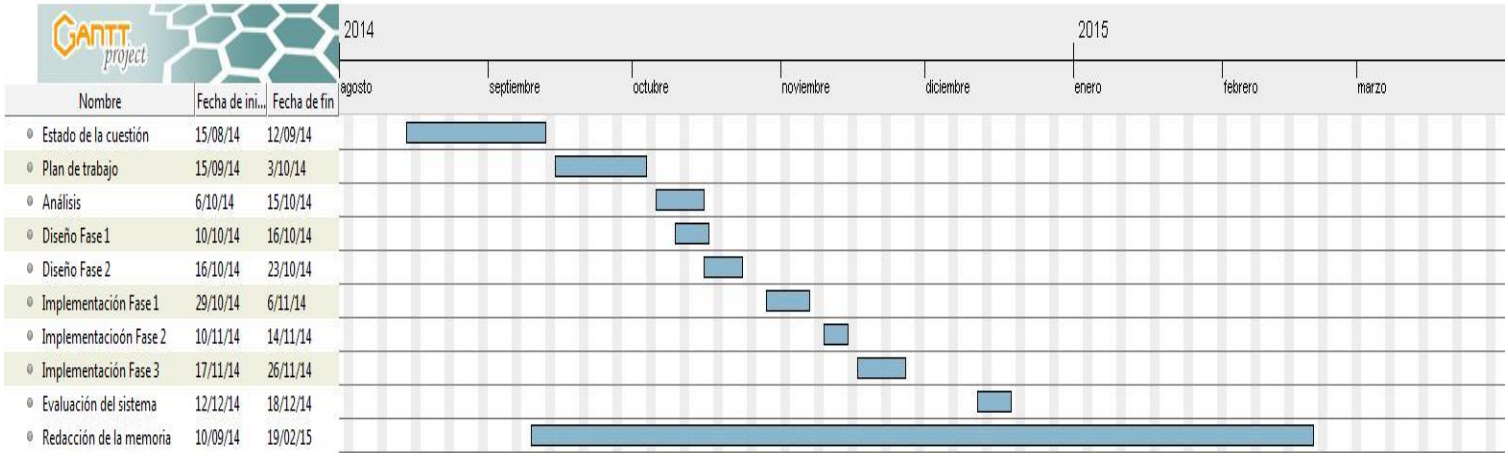
La forma de seguimiento que se ha llevado a cabo se ha basado en diferentes reuniones establecidas con el tutor y la directora del proyecto. Estas reuniones se realizaban de forma quincenal desde los meses de Septiembre del 2014 y Enero del 2015, siempre con motivo de mejorar el presente documento o implementación del proyecto. Es decir, este tipo de reuniones tienen como finalidad comprobar el estado del proyecto, si se cumplen los plazos, los hitos a cumplir desde la anterior reunión y mejorar la calidad de la memoria incluyendo nuevos conceptos o perspectivas para dar otro punto de vista con el fin de mejorar la calidad final del producto.

Las reuniones han sido muy útiles cuando se habla de la implementación del código, ya que diferentes problemas encontrados no se solucionaban con los conocimientos del alumno, por lo que era necesario asistir a estas reuniones y así avanzar en la implementación del código

Planificación del TFG

La planificación del proyecto fin de grado se ha formalizado mediante un diagrama de Gantt donde se especifican todas las tareas que se han realizado a lo largo de los meses de trabajo junto con las fechas de duración de las mismas.

La siguiente ilustración muestra la planificación:



A continuación se describen cada una de las tareas indicadas en el diagrama anterior:

- **Estado de la cuestión:** Se debe establecer el contexto del proyecto a desarrollar y el problema a resolver. Este apartado finaliza con un estudio exhaustivo de las tecnologías actuales que pueden compararse con la aplicación que se desea desarrollar para así poder encontrar los puntos que las diferencian o incluso mejorarlos.
- **Plan de trabajo:** Se identifican las tareas a realizar, un estudio inicial del problema a tratar y una estimación/planificación de las tareas expuestas anteriormente.
- **Análisis:** Una de las tareas más importantes en la fase de implementación donde se debe especificar el sistema que se plantea construir para resolver el problema. Además se establecen todas las necesidades y condiciones que debe cumplir el sistema que se desea desarrollar, que se plasmarán a modo de requisitos.
- **Diseño Fase 1:** Los requisitos expuestos en la fase anterior sirven para definir la arquitectura, funcionalidad y calidad del producto. En esta fase tiene como objetivo guiar el proceso de producción del sistema, generando modelos de entidades que se construirán en la siguiente fase.
- **Diseño Fase 2:** Se denomina al diseño detallado de la aplicación, donde se muestra el diseño final con todo detalle y con una breve explicación del funcionamiento de la misma
- **Implementación Fase 1:** Puesta a punto del entorno de trabajo para el desarrollo de la aplicación. Incluye interacción entre Hadoop, Eclipse y Windows.
- **Implementación Fase 2:** Desarrollo de la parte de manejo de ficheros con Hadoop y su sistema de ficheros HDFS
- **Implementación Fase 3:** Lectura de ficheros .csv dentro del entorno Hadoop
- **Implementación fase 4:** Desarrollo del paradigma MapReduce
- **Evaluación del sistema:** Se exponen los procesos que se han llevado a cabo para comprobar que la aplicación tiene un comportamiento correcto y cumple su funcionalidad.
- **Redacción de la memoria:** Elaboración del presente documento

Anexo III. Requisitos

Identificador	RF-1		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	La ventana principal del interfaz debe comprender dos secciones claramente diferenciadas: Listado Hadoop y visualización TextBox con los campos de los ficheros		
Prueba asociada	1. Arrancar la aplicación 2. Observar la ventana principal		
Caso de uso asociado	CU-3		

Tabla 44: Requisito Funcional 1

Identificador	RF-2		
Prioridad	ALTA	Necesidad	ESENCIAL
Descripción	El usuario no dispondrá de ningún fichero en HDFS hasta que no se realice la primera carga de ficheros		
Prueba asociada	1. Arrancar la aplicación 2. No realizar ninguna carga previa al sistema de ficheros HDFS 2. Comprobar que el listado Hadoop está vacío en la ventana principal de la aplicación		
Caso de uso asociado	CU-3		

Tabla 45: Requisito Funcional 2

Identificador	RF-3		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	El botón “File” abre un nuevo navegador (File Search) para interactuar con el árbol de ficheros del propio disco duro interno		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search		
Caso de uso asociado	CU-1		

Tabla 46: Requisito Funcional 3

Identificador	RF-4		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	File Search dispone de un árbol de ficheros del disco duro y una previsualización del fichero seleccionado		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. La ventana se compone de un árbol de ficheros C:\\ y un Jframe donde se visualizará el fichero seleccionado		
Caso de uso asociado	CU-1		

Tabla 47: Requisito Funcional 4

Identificador	RF-5		
Prioridad	MEDIA	Necesidad	OPCIONAL
Descripción	Los botones de acceso en File Search solo estarán disponibles si se selecciona un fichero del árbol		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Los botones de la ventana solo estarán disponibles cuando se selecciona un fichero del árbol de ficheros		
Caso de uso asociado	CU-1		

Tabla 48: Requisito Funcional 5

Identificador	RF-6		
Prioridad	ALTA	Necesidad	ESENCIAL
Descripción	Cuando se selecciona un fichero en File Search y se clica sobre el botón “LOAD”, el fichero se copiará a la ruta preestablecida de Hadoop como un fichero HDFS		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar un fichero .csv del árbol de ficheros 5. Clicar sobre el botón LOAD		
Caso de uso asociado	CU-1		

Tabla 49: Requisito Funcional 6

Identificador	RF-7		
Prioridad	ALTA	Necesidad	ESENCIAL
Descripción	El fichero copiado en File Search debe estar disponible en el listado		

	Hadoop del interfaz principal de la aplicación
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar un fichero .csv del árbol de ficheros 5. Clicar sobre el botón LOAD 6. Comprobar que el fichero cargado se encuentra en el listado Hadoop de la ventana principal del programa
Caso de uso asociado	CU-1

Tabla 50: Requisito Funcional 7

Identificador	**RF-8**		
Prioridad	MEDIA	**Necesidad**	OPCIONAL
Descripción	El botón "View" de File Search permite visualizar parte del contenido del fichero seleccionado en el explorador de ficheros de Windows. La previsualización estará disponible en el cuadro de texto de la parte derecha del navegador File Search		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar un fichero .csv del árbol de ficheros 5. Clicar sobre el botón VIEW 6. El contenido del fichero debe mostrarse parcialmente en el JFrame de la parte derecha de la ventana File Search		
Caso de uso asociado	CU-3		

Tabla 51: Requisito Funcional 8

Identificador	**RF-9**		
Prioridad	MEDIA	**Necesidad**	DESEABLE
Descripción	Cuando se previsualiza un fichero en File Search se habilitará el botón de "Show me more…"		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar un fichero .csv del árbol de ficheros 5. Clicar sobre el botón VIEW 6. El contenido del fichero debe mostrarse parcialmente en el JFrame de la parte derecha de la ventana File Search 7. El botón Show me more… ahora está habilitado		
Caso de uso asociado	CU-3		

Tabla 52: Requisito Funcional 9

Identificador	RF-10		
Prioridad	MEDIA	Necesidad	DESEABLE
Descripción	El botón “Show me more” permite abrir el bloc de notas con el contenido total del fichero seleccionado		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar un fichero .csv del árbol de ficheros 5. Clicar sobre el botón VIEW 6. El contenido del fichero debe mostrarse parcialmente en el JFrame de la parte derecha de la ventana File Search 7. Clicar el botón Show me more 8. Se abre el bloc de notas con el contenido completo del fichero seleccionado		
Caso de uso asociado	CU-3		

Tabla 53: Requisito Funcional 10

Identificador	RF-11		
Prioridad	MEDIA	Necesidad	DESEABLE
Descripción	El botón “Reload” de File Search permite refrescar el explorador de ficheros de Windows y volver a la posición inicial desde el principio. Debe estar disponible en cualquier momento		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Clicar el botón Refresh 5. El árbol de ficheros C:\\ debe actualizarse		
Caso de uso asociado	CU-1		

Tabla 54: Requisito Funcional 11

Identificador	RF-12		
Prioridad	BAJA	Necesidad	DESEABLE
Descripción	Pulsando nuevamente sobre el botón “File” de la barra de herramientas se cerrará el navegador de File Search		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Seleccionar nuevamente el botón File 5. La ventana File Search se cierra para volver a la ventana principal		
Caso de uso asociado	CU-1		

Tabla 55: Requisito Funcional 12

Identificador	RF-13		
Prioridad	BAJA	Necesidad	OPCIONAL
Descripción	Es posible cerrar el navegador File Search mediante el aspa ubicada en la parte superior derecha del mismo		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar el botón File de la barra de herramientas superior 3. Se abre la ventana de File Search 4. Cerrar la nueva ventana mediante el icono rojo con aspa en la parte superior derecha 5. La ventana File Search se cierra para volver a la ventana principal		
Caso de uso asociado	CU-1		

Tabla 56: Requisito Funcional 13

Identificador	RF-14		
Prioridad	ALTA	Necesidad	ESENCIAL
Descripción	Cuando seleccionamos un fichero del listado principal de Hadoop en el interfaz principal y clicamos sobre el botón “Delete File”, el fichero se elimina del listado y de la ubicación Hadoop establecida.		
Prueba asociada	1. Arrancar la aplicación 2. Seleccionar un fichero del listado Hadoop 3. Clicar el botón Delete 4. El fichero eliminado ya no debe aparecer en el listado Hadoop		
Caso de uso asociado	CU-2		

Tabla 57: Requisito Funcional 14

Identificador	RF-15		
Prioridad	BAJA	Necesidad	DESEABLE
Descripción	Cuando se selecciona el botón “MapReduce” de la barra de herramientas, se abrirá un nuevo navegador denominado MapReduce		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce		
Caso de uso asociado	CU-4		

Tabla 58: Requisito Funcional 15

Identificador	RF-16		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	En el navegador MapReduce se establecen tres listados: Files, Key/s y Value/s		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales		
Caso de uso asociado	CU-4		

Tabla 59: Requisito Funcional 16

Identificador	RF-17		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	El listado Files del navegador MapReduce contiene todos los ficheros cargados en la ubicación Hadoop y son seleccionables		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado)		
Caso de uso asociado	CU-4		

Tabla 60: Requisito Funcional 17

Identificador	RF-18		
Prioridad	MEDIA	Necesidad	DESEABLE
Descripción	Los listados Key/s y Value/s estarán deshabilitados hasta que no se seleccione un fichero del listado Hadoop		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado		
Caso de uso asociado	CU-5		

Tabla 61: Requisito Funcional 18

Identificador	RF-19		
Prioridad	BAJA	Necesidad	DESEABLE
Descripción	Cuando seleccionamos un fichero Hadoop del navegador MapReduce se habilitan los listados Key/s y Value/s		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Se habilita el listado Keys y Values según se vaya seleccionando del listado		
Caso de uso asociado	CU-5		

Tabla 62: Requisito Funcional 19

Identificador	RF-20		
Prioridad	MEDIA	Necesidad	ESENCIAL
Descripción	Los listados Key/s y Value/s del navegador MapReduce contienen todos los campos estructurados del fichero seleccionado		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Se habilita el listado Keys y Values según se vaya seleccionando del listado 6. Los listados tienen que ser idénticos en valores de los listados y deben ser los campos de la estructura del propio fichero		
Caso de uso asociado	CU-4		

Tabla 63: Requisito Funcional 20

Identificador	RF-21		
Prioridad	BAJA	Necesidad	DESEABLE
Descripción	Es posible seleccionar más de un valor en los listados Key/s y Value/s del navegador MapReduce		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Se habilita el listado Keys y Values según se vaya seleccionando del listado 6. Con la tecla ctrl+click sobre los listados keys y values es posible seleccionar más de un valor del listado.		
Caso de uso asociado	CU-5		

Tabla 64: Requisito Funcional 21

Identificador	RF-22		
Prioridad	BAJA	Necesidad	DESEABLE
Descripción	Una vez se selecciona el fichero, la key/s y el value/s de un fichero Hadoop en el navegador MapReduce, es posible elegir entre las opciones “All Files” y “Selected File”		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Seleccionar keys y values de sus respectivas listas 6. Se ha habilitado los radiobuttons para la selección de un fichero o todos los ubicados en el directorio Hadoop		

Caso de uso asociado	CU-5

Tabla 65: Requisito Funcional 22

Identificador	**RF-23**		
Prioridad	BAJA	**Necesidad**	DESEABLE
Descripción	Cuando se selecciona cualquier radioButton del navegador MapReduce se habilita el botón “MapReduce”		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Seleccionar keys y values de sus respectivas listas 6. Seleccionar un radioButton de las dos opciones disponibles 7. Se habilita el botón “Map Reduce”		
Caso de uso asociado	CU-4		

Tabla 66: Requisito Funcional 23

Identificador	**RF-24**		
Prioridad	MEDIA	**Necesidad**	ESENCIAL
Descripción	Si se clica sobre el botón “MapReduce” del navegador MapReduce saltará un mensaje para visualizar o no el fichero generado por el paradigma MapReduce		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Seleccionar keys y values de sus respectivas listas 6. Seleccionar un radioButton de las dos opciones disponibles 7. Clicar el botón “Map Reduce” 8. Cuando finalice el proceso se mostrará un mensaje para abrir el		

	fichero con los resultados si se desea
Caso de uso asociado	CU-4

Tabla 67: Requisito Funcional 24

Identificador	**RF-25**		
Prioridad	MEDIA	**Necesidad**	OPCIONAL
Descripción	Cuando se selecciona visualizar el fichero generado por el paradigma MapReduce, se abrirá el bloc de notas para visualizar el resultado obtenido por el paradigma.		
Prueba asociada	1. Arrancar la aplicación 2. Clicar el botón MapReduce de la barra de herramientas superior 3. Se abre una nueva ventana llamada MapReduce con tres listados principales 4. Solo está disponible el listado File para seleccionar un fichero. Los otros dos listados aparecen deshabilitados hasta que no se seleccione del primer listado 4. Seleccionar un fichero de la lista File (previamente se ha tenido que cargar un fichero a la ruta Hadoop para que esté disponible en este listado) 5. Seleccionar keys y values de sus respectivas listas 6. Seleccionar un radioButton de las dos opciones disponibles 7. Clicar el botón “Map Reduce” 8. Seleccionar YES en la ventana de resultado MapReduce para visualizar el fichero resultado 8. Se abre el bloc de notas con el resultado arrojado por MapReduce		
Caso de uso asociado	CU-4		

Tabla 68: Requisito Funcional 25

Identificador	**RNF-1**		
Prioridad	ALTA	**Necesidad**	ESENCIAL
Descripción	El entorno sólo podrá ser operativo si se cuenta con una instalación del framework Hadoop		
Prueba asociada	1. Arrancar la aplicación 2. Si aparece el error “Retrying Access Hadoop” no se cuenta con la instalación correcta de Hadoop		

Tabla 69: Requisito No Funcional 1

Identificador	**RNF-2**		
Prioridad	ALTA	**Necesidad**	ESENCIAL
Descripción	Es necesario disponer de las librerías Hadoop necesarias para lanzar la		

	aplicación
Prueba asociada	1. Arrancar la aplicación 2. Si aparece el error “Unable to acces awg.Hadoop” no se cuenta con la librería necesaria para lanzar cualquier tipo de acceso a Hadoop

Tabla 70: Requisito No Funcional 2

Identificador	**RNF-3**		
Prioridad	BAJA	**Necesidad**	DESEABLE
Descripción	El estándar aplicado al entorno visual es el X-Windows de uso generalizado		
Prueba asociada	1. Arrancar la aplicación 2. El aspecto visual de la aplicación debe ser parecido a Windows, con los botones de minimizar, maximizar y cerrar en la barra superior de cualquier ventana del programa		

Tabla 71: Requisito No Funcional 3

Identificador	**RNF-4**		
Prioridad	MEDIA	**Necesidad**	ESENCIAL
Descripción	La aplicación estará optimizada para cualquier versión de Windows actual (a partir de Windows XP)		
Prueba asociada	1. Arrancar la aplicación sobre cualquier instalación Windows que se desee 2. La aplicación arranca		

Tabla 72: Requisito No Funcional 4

Identificador	**RNF-5**		
Prioridad	MEDIA	**Necesidad**	DESEABLE
Descripción	La aplicación solo estará disponible con el idioma Ingles		
Prueba asociada	1. Arrancar la aplicación 2. Comprobar que los literales de la aplicación aparecen únicamente en el idioma ingles		

Tabla 73: Requisito No Funcional 5

Identificador	**RNF-6**		
Prioridad	BAJA	**Necesidad**	DESEABLE
Descripción	La aplicación está pensada para una resolución mínima de 1024x768 pixeles		
Prueba asociada	1. Arrancar la aplicación sobre una resolución mínima de 1024x768 2. Si la resolución no es la adecuada, la aplicación no podrá verse por completo sobre la pantalla del monitor		

Tabla 74: Requisito No Funcional 6

Identificador	RNF-7		
Prioridad	MEDIA	Necesidad	DESEABLE
Descripción	La proporción de los tamaños de los elementos debe ser estática y no se dimensionará según la resolución de pantalla		
Prueba asociada	1. Arrancar la aplicación sobre una resolución mínima de 1024x768 2. Si la resolución no es la adecuada, la aplicación no podrá verse por completo sobre la pantalla del monitor 3. No es posible redimensionar el tamaño de las ventanas de la aplicación		

Tabla 75: Requisito No Funcional 7

Identificador	RNF-8		
Prioridad	ALTA	Necesidad	ESENCIAL
Descripción	Solo estará disponible un resultado del paradigma MapReduce. Si el usuario lanza nuevamente el paradigma, se borrará el anterior resultado obtenido		
Prueba asociada	1. Arrancar la aplicación 2. Clicar sobre el botón MapReduce de la barra de herramientas superior 3. Elegir el fichero que se desee, sus keys y values y el radioButton de ficheros 4. Clicar sobre el botón MapReduce de la ventana 5. Comprobar la carpeta C:\\hdp-folders\output . Solo debe existir un fichero part-SUCESS 6. Procesar otro paradigma MapReduce 7. Comprobar la carpeta C:\\hdp-folders\output . Solo debe existir un fichero part-SUCESS		

Tabla 76: Requisito No Funcional 8

Identificador	RNF-9		
Prioridad	MEDIA	Necesidad	DESEABLE
Descripción	No se guardará ningún historial referente a la anterior ejecución del programa		
Prueba asociada	1. Arrancar la aplicación 2. Realizar cualquier tipo de acción en MapReduce		

	3. Cerrar la aplicación 4. Arrancar nuevamente la aplicación 5. Comprobar que no se ha guardado referencias a la última conexión en la aplicación

Tabla 77: Requisito No Funcional 9